VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

RASPBERRY PI: PROGRAMOVÁNÍ V PROSTŘEDÍ MATLAB/SIMULINK
RASPBERRY PI: PROGRAMMING BY MEANS OF MATLAB/SIMULINK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Tomáš Holoubek**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Radomil Matoušek, Ph.D.**
SUPERVISOR

BRNO 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Tomáš Holoubek**
Studijní program:	Strojírenství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**doc. Ing. Radomil Matoušek, Ph.D.**
Akademický rok:	2016/17

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Raspberry Pi: programování v prostředí Matlab/Simulink

Stručná charakteristika problematiky úkolu:

S využitím prostředí Matlab/Simulink (MATLAB® Support Package for Raspberry Pi® Hardware) budou vytvořeny minimálně tři demonstrační úlohy pro vývojový kit Raspberry Pi 3. Platforma Raspberry s příslušenstvím i sw jsou k dispozici.

Cíle bakalářské práce:

1) Seznámení s problematikou programování platformy Raspberry Pi.
2) Seznamení s MATLAB® Support Package for Raspberry Pi® Hardware.
3) Po konzultaci se školitelem navrhnout minimálně tři demonstrační úlohy, pro které bude rovněž vytvořen popis.
4) Realizace úloh.

Seznam literatury:

UPTON Eben; HALFACREE Gareth: Raspberry Pi, COMPUTER PRESS, 2016. Martin Stříž, Bučovice, 2015. 280 s. ISBN: 978-80-251-4819-8

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2016/17.

V Brně, dne 7. 11. 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ
DĚKANÁT

L. S.

doc. Ing. Radomil Matoušek, Ph.D.
ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
děkan fakulty

ABSTRAKT

Cílem této práce je seznámení s Raspberry Pi, instalace hardwarové podpory, základní popis prostředí a programování v Matlab/Simulink. Vytvoření demonstračních příkladů s využitím stereo mikrofonu, kamery a mikro servomotorů.

ABSTRACT

The Goal of this thesis is an introduction to Raspberry Pi, specifically an installation of hardware support and basic description of setting and coding using Matlab/Simulink. Creation of applicative examples by using stereo microphones, camera and micro sized servo motor is demonstrated.

KLÍČOVÁ SLOVA

Raspberry Pi, programování v Matlab, Simulink, určení směru zdroje zvuku, sledování objektu, kalibrace, reálné souřadnice, reálná velikost objektu, PWM, ovládání serva v Simulinku, 3D polohování kamery, sférické souřadnice.

KEYWORDS

Raspberry Pi, coding using Matlab, Simulink, DOA, direction of arrival, object tracking, calibration, factual coordinates, determination of size of an object, PWM, controlling the servo motor using Simulink, 3D camera adjustment, spherical coordinates.

BIBLIOGRAFICKÁ CITACE

HOLOUBEK, T. *Raspberry Pi: programování v prostředí Matlab/Simulink*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2017. 84 s. Vedoucí bakalářské práce doc. Ing. Radomil Matoušek, Ph.D

PODĚKOVÁNÍ

Rád bych poděkoval panu doc. Ing. Radomilu Matouškovi, Ph.D. za odborné vedení, ochotu a cenné rady, které mi pomohly při tvorbě této práce.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením pana doc. Ing. Radomila Matouška, Ph.D. a s použitím literatury uvedené v seznamu literatury.

V Brně dne 24. 5. 2017 ……

Tomáš Holoubek

OBSAH

1 ÚVOD

V současnosti výkonných počítačových sestav a mobilních technologií se vývoj začíná stále více zaměřovat na miniaturizaci a využití výpočetního výkonu v různých embedded typech řešení, které jsou energeticky nenáročné s velmi příznivou cenou. To přináší zcela nové možnosti a rozšiřuje dosavadní způsob užívání novým směrem, jako je např. rostoucí zájem o IoT (*Internet of Things*).

Jedním z těchto zařízení je snad nejznámější miniaturní počítač Raspberry Pi, který se již od první verze stal velice rychle populárním nejen pro své vlastnosti, ale i z důvodů snadného propojení a ovládání periferií, možností integrace a v podstatě neomezenými možnostmi vytvořit na jeho základu výsledný projekt jakéhokoliv způsobu využití.

Tato práce se nejdříve zabývá seznámením s Raspberry Pi a popisem možností tohoto mini počítače. Součástí je výpis porovnání vlastností s podobnými produkty, které jsou na trhu k dispozici. Cílem práce je prezentovat možnosti využití Raspberry Pi ve spojení s programem Matlab a jeho nadstavbou Simulink, jež jsou již určitým standardem pro technické a vědecké výpočty. Tento program disponuje obrovským potenciálem a jeho výpočetní algoritmy ve spojení i s relativně málo výkonným zařízením dosahují velké efektivity.

Následně jsou zde popsány 3 demonstrační příklady s podrobným rozborem a vysvětlením včetně použitých schémat a zdrojových kódů.

První příklad obsahuje způsob řešení pro výpočet odhadu směru příchozího zvuku, kde je použita externí zvuková karta a stereomikrofon vlastní výroby. Připojeno je řešení s použitím bloku Simulinku přímo určeného pro tento typ úlohy.

Druhý demonstrační příklad se zabývá detekcí a sledováním objektu při živém snímání kamerou s využitím metody segmentace barev. Součástí je kalibrace, která umí určit poměr zobrazení, vypočte reálné souřadnice s rozměry sledovaného objektu a stanoví vzdálenost pozadí od kamery.

Poslední třetí příklad o polohování kamery v prostoru rozebírá problematiku připojení a ovládání externího zařízení přes Raspberry Pi. Pro tento účel byl vyroben model 3D polohovací konstrukce. Chybějící podpora pro ovládání serva v Simulinku je řešena vytvořením vlastní funkce pro řízení PWM signálu. Doplňkem je přepočet lokální polohy objektu na obrazovce v kombinaci s otáčením kamery na globální sférický souřadnicový systém.

2 RASPBERRY PI

Tato kapitola je seznámením s platformou mini počítače Raspberry Pi.

2.1 Historie Raspberry Pi

Chabé znalosti a nezájem nové generace mladých studentů o programování a informatiku přivedla učitele jménem Eben Upton z Cambridgeské univerzity na poměrně revoluční nápad, jak tuto situaci změnit.

Eben Upton a jeho hlavní kolegové dr. Rob Mullins a prof. Alan Mycroft se společně ještě s několika dalšími počítačovými nadšenci z univerzitní laboratoře rozhodli v tomto projektu spojit své síly a jak popisují, založili malou nadaci s velkými plány *„a little charity with big ideas“* [2, s. 13] s názvem Raspberry Pi Foundation.

V roce 2005 se původní myšlenka začala měnit na konkrétní reálný projekt. Cílem bylo vytvořit malý jednoduchý počítač, který by pomohl při výuce základů programování a neměl by být výrazně dražší než učebnice. K tomuto mini pc se pak připojí monitor nebo TV, klávesnice a pevný disk nahradí paměťová karta.

Zásadním faktorem byla v té době dostupnost výkonného hardware, zejména čipů od společnosti Broadcom, kterým disponovaly mobilní telefony a měly již v sobě integrovanou například pamět, grafiku 3D a video v HD.

Koncem roku 2011 vznikla první verze jednoduchého miniaturního počítače velikosti kreditní karty, zároveň dostatečně výkonného a hlavně cenově dostupného. Přes počáteční nesnáze s rozvojem a propagací přes společnost BBC bylo nakonec vytvořeno prezentační video, které v podstatě obletělo svět a stalo se jedním ze startovacích bodů ke komerčnímu rozšíření.

29. února 2012 byl zahájen oficiální prodej a první verze Raspberry Pi A měla předběžně stanovenou cenu na pouhých $ 25 za základní verzi.

Tento mini pc prezentovaný jako nejmenší počítač na světě dosáhl ihned obrovského úspěchu. Pod náporem zájemců oficiální web několikrát zkolaboval, výroba nestíhala, prodáno bylo statisíce kusů a ke konci roku 2013 počet prodaných kusů dosáhl 2 mil. Následovaly další vylepšené verze a zájem nepřestává růst až do současnosti. Tato společnost zrodila komunitu nadšenců, výukový web, vydává např. MagPi časopis atd.

Raspberry Pi zvládne cokoliv co běžný počítač, lze ho použít na grafiku, videa, multimédia (např. multimediální centrum), jako herní platformu, pro programování her, pro zábavu obecně - k čemuž byl vlastně i určen - pro zábavu a hraní spojené s učením programování a vzděláváním.

To co ale dělá tento mini pc opravdu zajímavým, jsou praktické možnosti programování zařízení, ovládání jednoduchých i složitých systémů od sběru dat, ovládání domácnosti mobilním tel., multimediálního centra, nebo např. pohyb vozíku s akváriem, jež ovládá sama rybka až po určitou automatizaci a složité autonomní systémy, či některé výrobní procesy. [1, 2, 3]

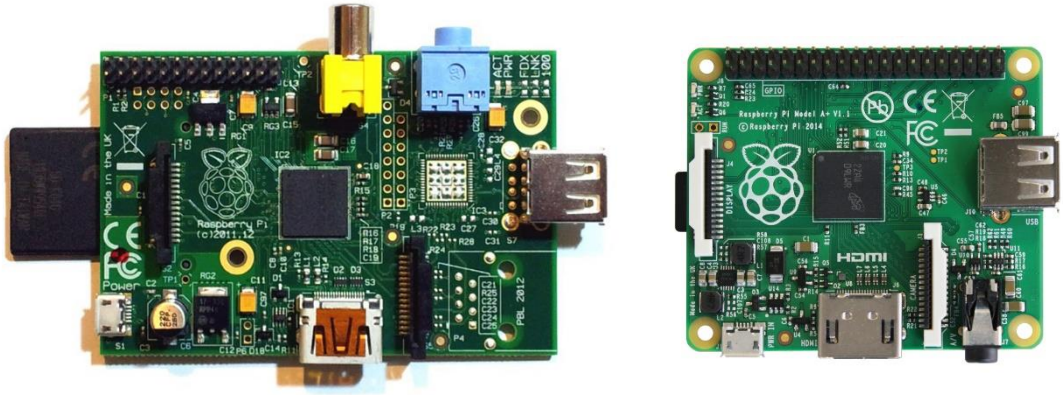
2.2 Modely Raspberry Pi

RASPBERRY PI 1

Úplně první prodávaný typ byl **MODEL A**. Základní výbavou byl procesor Broadcom BCM2835 SoC s kmitočtem 700 MHz, GPU VideoCore IV, A/V výstup.
První verze začínala s RAM 256 MB a jedním USB portem, paměť byla sdílená společně s grafickou kartou. V modelech A chyběl LAN port. Tento základ nedostačoval a byl poměrně rychle nahrazen v listopadu 2014 novým modelem A+ .[1, 2, 5]

MODEL A+ dostal větší RAM 512 MB, konektor A/V (kompozitní video výstup) změněn na 3,5mm jack, přidán nově HDMI konektor. Počet GPIO pinů zvýšen z 26 na 40. Díky tomu se objevilo na internetu mezi uživateli tou dobou snad nejvíce návodů na různá vylepšení pro projekty.

Dále spotřeba energie snížena na rozmezí od 0,5 až 1,0 W a rozměr na délku celé desky klesl cca o třetinu (21mm). Konektorem napájení nově microUSB.
Významnou změnou byla výměna SD klasické paměťové karty za microSD, jak lze vidět na levé straně u obr. 1) , slot pro microSD přemístěn na spodní stranu.

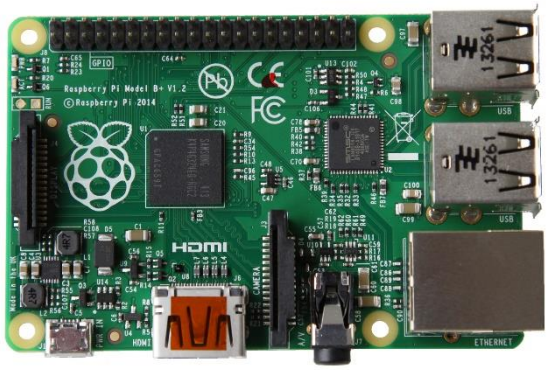


Obr. 1: Raspberry Pi, zleva model A, model A+ [8, 9]

Novější vydání modelu B přišlo do prodeje již začátkem 2012. Důležitým vylepšením bylo přidání 2. portu USB a také konečně dlouho očekávaného konektoru RJ45 pro ethernet 10/100. Pamět (sdílená s grafickou kartou) zůstala na původní hodnotě 512 MB. [3, 6, 7]

Model B+ (viz Obr 2.) se objevil až za další 2 roky v polovině 2014 a změny byly poznat na první pohled. Sice zůstal původní procesor Broadcom běžící na 700 MHz, ale pamět byla zvětšena na 1 GB. Dále bylo vyhověno uživatelům, kteří stále žádali další USB porty, nově tedy místo 2 původních nyní 4x USB 2.0. Změna slotu z SD na microSD kartu se již stal standardem, ale vylepšena na push-push verzi (stlačením vložit a vysunout). Zdroj 5 V s ochranou proti přepólování, 2 A pojistka. U audia díky novým obvodům došlo ke snížení šumu. Dále dostal nový řadič pro Ethernetový čip.

Grafika má rychlé 3D jádro s přístupem k OpenGL ES2.0 a OpenVG knihovně.
Z dosavadních modelů byl nejrozšířenějším a nejpoužívanějším typem. [3, 11, 12, 13]



Obr. 2: Raspberry Pi B+ [11]

RASPBERRY PI 2

Druhá generace začla rovnou označením B, tedy **Raspberry Pi 2 B** a nahradila předchozí model 1 B+ v roce 2015. Vizuálně se příliš nelišil. Novinkou bylo osazení desky novým 4-jádrovým procesorem 900MHz quad-core ARM Cortex-A7. Ostatní parametry zůstaly stejné jako u předchozího modelu. Přesto chybělo ještě něco málo (např. Wi-Fi modul), to bylo pak splněno později dalším typem 3. generace: Raspberry Pi 3 Model B. Podrobný popis a specifikace jsou uvedeny v následnující kapitole. [13, 18]

RASPBERRY PI ZERO

Společnost Raspberry Pi Foundation se na chvíli vrátila ke své původní myšlence stvořit opravdu velmi levný a malý počítač a překvapila uvedením nejmenšího ze všech dosavadních pc.

Obr. 3: Raspberry Pi Zero W [17]

Raspberrry Pi Zero má rozměry 65 x 30 x 5 mm a cenu $ 5. Na délku měří stejně jako model A+, ale na šířku jen 60 % jeho velikosti. Parametry jsou základní, jde o známý jednojádrový 1 GHz ARM a 512 MB RAM. Velikost microSD je zde nutností, dále mini HDMI (1080p60), konektor na kameru zůstal, GPIO piny lze přidat do připravených kontaktů dle potřeby (max 40), stejně tak RCA kompozitní video.

Hned další verze byla ZERO W (viz obr. 3) s písmenem „W“ značícím Wi-Fi. Bohužel, slabou stránkou je absence běžných USB, nahrazují je 2 porty microUSB. Řešení spočívá v adaptérech (kabel s redukcí) nebo přídavném modulu. Ještě jedno malé

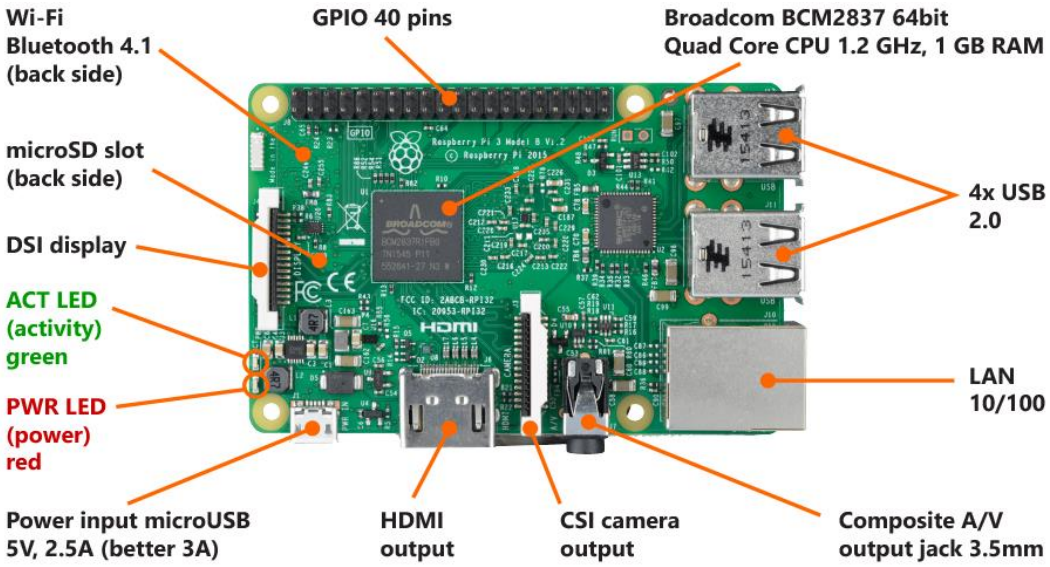
mínus je v současnosti stálá prodejní nedostupnost. Velikost tohoto pc z něj ale dělá ideálního kandidáta pro zabudování do menších konzolí, modelů aut, dronů apod. [14, 15, 16]

2.3 Popis a parametry Raspberry Pi 3 Model B

To je v současnosti poslední a stále nejnovější model, který je i hlavním hardwarem v této práci. V únoru 2016 nahradil model 2B a ve srovnání s ním je novinkou:

- A 1.2GHz 64-bit Quad Core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1 Classic, Bluetooth Low Energy (BLE)

Ostatní stejné jako u Pi 2: 1 GB RAM, 4x USB, 40 GPIO pinů, Full HDMI port, Ethernet port (RJ45), výstup A/V 3.5mm audio jack, Camera interface (CSI), Display interface (DSI), MicroSD (typ push-pull místo push-push), VideoCore IV 3D grafické jádro. Konektor napájení micro USB s doporučením na adaptér 2,5A kvůli pravděpodobnému vytížení USB konektorů. Bohužel stále nemá řízení reálného času (RTC).



Obr. 4: Raspberry Pi 3 Model B

Jeho nový 4-jádrový 64-bitový procesor je 10x výkonnější než čip první verze, ovšem taky dosahuje vysokých teplot (cca 80°C) při plné zátěži a pasivním chlazení. I když se při každodenním standardně vytíženém provozu teplota drží v normálu bez výrazného překročení průměru a nenastává obávané přehřátí *(overheating)*, nedoporučuje se úplné uzavření RPi do krabičky s minimálním větráním.

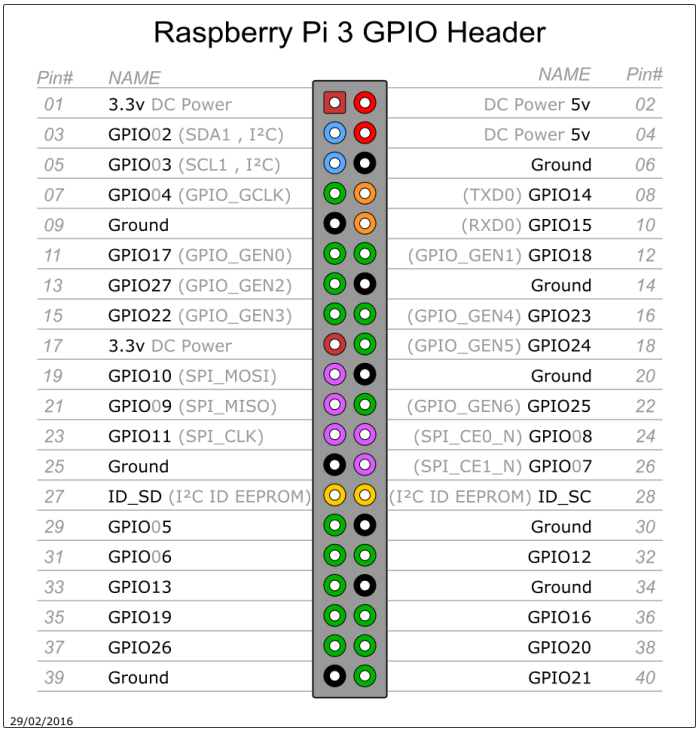
Integrovaná Wi-Fi znamená konec nepraktickému připojování externích adaptérů (Wi-Fi dongle). Kompatibilita se staršími modely Raspberry Pi 1 a 2 má být bezproblémová. [18, 19, 20, 21, 42]

Pro celou další práci je použit tento typ mini pc. Je hlavní součástí hardwaru pro praktické úkoly se softwarem Matlab a Simulink.

2.4 GPIO

GPIO (General Purpose Input/Output) je hardwarové rozhraní vstupu/výstupu, které je integrované na desce a je ve výchozím stavu vypnuté, přičemž počet určený k dispozici může uživatel za provozu libovolně softwarově ovládat. Do těchto pinů lze posílat el. signál a ovládat tak jiný hardware na nejnižší úrovni. Lze na nich nastavit 0V/3,3V jako logickou 0/1. Velkou výhodou je, že není nutné žadné nastavování.

Pro Raspberry Pi 3 B jde celkem o 40 pinů (viz obr. 5), obsahuje rozhraní UART, sběrnici I2C, SPI, I2S audio, 3V3 (3,3V), 5V a zem. Maximální počet pinů lze teoreticky rozšiřovat do nekonečna. [3, 23]



Obr. 5: Raspberry Pi 3 B popis GPIO [25]

Zároveň lze použít některý z prodávaných již většinou kompletních připravených modulů (shield, HAT) a připojit ho přímo na GPIO. Tyto moduly pak rozšiřují možnosti mini pc a proměňují ho v mocný nástroj. Přes GPIO lze ovládat jednoduchá zařízení s diodami, tlačítka, mikrokontrolery po internetu, získávání dat z čidel a další.

Zapojování je obecně jednoduché, přesto je tu nebezpečí spálení celé desky. Zařízení, které po připojení pustí přes GPIO piny větší napětí než povolených 3,3V, velmi pravděpodobně Rasberry Pi zničí. Elektronika není tolerantní k vyššímu napětí a nemá ochranu přepětí na zvládnutí přechodu od 5V na 3,3V, je totiž zapojena přímo na čip. Druhým problémem může být vysoký odběr, zapojení diody je ok, ale zapojení relé nebo

motoru už ne. Zde se nabízí použití některého shieldu, který nutnou ochranu zajistí. [22, 23, 24]

2.5 Konkurence

Popularita a velké prodeje Raspberry Pi probudila konkurenci k činnosti a brzo se v této kategorii objevují modely od jiných společností, které především využívají nedostatků Rasberry Pi („RPi“) a snaží se nabídnout něco navíc nebo něco netradičního.

Banana Pi
V podstatě klon Raspberry Pi, navíc je SATA řadič, mikrofon, IR přijímač, tlačítko reset a power, podpora SATA disků až do velikosti 2 TB [26].

BeagleBone Black (BBB)
Nabízí 2x PRU 32-bit mikrokontroléry (Programable realtime unit), 4 GB on-board flash storage. Využití pro malou automatizaci nebo domácí CNC, 3D tiskárny [27, 28].

Orange Pi Plus
Čínský klon. Velmi dobrá výbava. Např. IR dálk. Ovládání, SATA 2.0, mikrofon [29].

PINE64
Podobná výbava za poloviční cenu. Např. IR, video 4K, 2G LPDDR3 RAM, Power Management Unit, Real Time Clock Battery port, power jack, podpora Androidu [30].

Arduino
Často porovnáváno, základním rozdílem je ale určení. RasPi je mini pc, Arduino je mikrokontroler bez podobného oper. systému. Výhodou je multiplatformost, open source, rozšiřitelnost. Jeho doménou je ovládání elektroniky, např otevírání a zavírání garážových vrat, zapínání světel, řízení mechaniky CNC, robotická paže. [31]

Další: ODROID-C2, JaguarBoard, UDOO, Tinker Board (od Asus), Lemon Pi.

2.6 Operační systémy pro Raspberry Pi

NOOBS (New Out Of the Box Software)
Jde o pomocný instalační program (systém), který velmi zjednodušuje výběr a instalaci oper. systému při prvním spuštění nového Raspberry Pi.

Není k dispozici žádný vestavěný HDD, operační systém se načítá a běží pouze z vložené microSD karty a jsou základní dvě možnosti, jak systém na tuto paměťovou kartu nainstalovat. Jednodušší verzí je koupě microSD karty s již připraveným NOOBS systémem. A nebo stáhnout z webu: https://www.raspberrypi.org/downloads/noobs/ *zip* soubor a po rozbalení systém vložit na levnější prázdnou kartu. Na zmíněném webu je na výběr ze dvou verzí NOOBS. Buď LITE verze, která kromě samotného nástroje neobsahuje žádný *image* soubor pro instalaci operačního systému, nebo plnohodnotný

NOOBS s instalací operačního systému Raspbian, který je nejpoužívanější. LITE verze se liší pouze tím, že po dobu instalace musí být RPi připojeno k internetu.
Pro stažení z webu je nutné mít microSD s min. kapacitou 8 GB (formát FAT32), po rozbalení *zip* souboru se obsah jednoduše nahraje na kartu.
Následně je postup stejný pro obě možnosti, vložit do Raspberry Pi a spustit - spustit znamená připojit napájecí adaptér do mini pc. Pokud je k dispozici připojení k netu, nabídka operačních systémů bude rozšířena o další možnosti, ale základní a doporučený Raspbian bude vždy k dispozici.

Tím, že je operační systém nahrán na microSD kartě, může být lehce vyměněn za jiný pouhým vysunutím karty a vložením jiné. [20, 32]

Nabídka operačních systémů v NOOBS
Je třeba zmínit, že změna v nastavení jazyka, klávesnice nebo displeje se po instalaci uloží do nového systému jako defaultní.
Raspbian - distribuce Linuxu, systém odvozený z Debianu, určen speciálně pro RPi.
Dále z těch hlavních například:
Windows 10 IoT Core - Windows 10 a jeho „internet věcí“, **Open ELEC**, **Pidora** a další [32, 33].
V této práci bude využita hw instalace systému podpory pro Matlab a Simulink, proto další podrobnosti o ostatních operačních systémech nejsou podrobněji rozebírány.

2.7 Periferie a příslušenství

Raspberry Pi je sice kompletní miniaturní pc, ale pro potřeby přímé komunikace při programování a ladění nebo jakéhokoliv provozu podobného klasickému pc potřebuje samozřejmě i ovládací zařízení. Tedy v základu monitor (stačí i TV), klávesnici a myš.
Samozřejmostí je i napájecí adaptér. Dále lze přímo na desku připojit kameru, dotykový display, externí zvukovou kartu, další USB zařízení, LAN (nebo je k dispozici Wi-Fi).

Množství příslušenství, které lze k RPi dokoupit, je dnes už velký počet. Výrobce přihlíží k zachování kompatibility od aktuálních až po starší modely.
V nabídce jsou prvky pasivního chlazení, krabičky, pro využití GPIO propojky, káblíky, nepájivá pole.

Pro vlastní RPi byl pořízen silnější 3 A napájecí adaptér místo doporučovaných 2,5 A, aby uživil kamerku dohromady se dvěma servomotory. Dále dokoupen velký hliníkový pasivní chladič, nízká krabička pro RPi s vrchním větracím otvorem pro přesahující chladič s lepším odvodem tepla

Přídavné moduly existují pro audio, kamery, displeje, řízení I/O, LED, senzory, reálný čas (RPi nemá řízení RTC !). Dále je tu výběr desek s přídavnými funkcemi (HAT) a připojitelné na GPIO jako je řízení mikrokontrolerů, PWM, relé, servomotorů. Přehled modelů i se zobrazením zapojení GPIO lze najít na: https://pinout.xyz/boards
[2, 3, 34, 35]

2.8 Možnosti programování Raspberry Pi

Vzhledem k tomu, že platforma byla vyvinuta za původním účelem přívést a motivovat k programování mladší generace, je součástí operačního systému Raspbian programovací prostředí podobné hře pro juniory s názvem Scratch. Tento jednoduchý výukový programovací jazyk je k dispozici ihned po zprovoznění systému při rozbalení nabídky v hlavní liště. Další variantou na stejném místě je Python ve 2 verzích s výchozím editorem IDLE a pár hotovými ukázkami. Je to jeden z hlavních programovacích jazyků, pro které je toto mini pc určeno, ovšem není problém se rozhodnout pro jazyk C, Pascal nebo cokoliv jiného.

Většina nachystaných doplňkových aplikací a modulů počítá prioritně s programováním v Pythonu. Např. existuje modul pro programování 3D a je k tomu pro tento jazyk připravena i knihovna.

Raspbian má i vlastní kancelářské programy. Je tu Abiword, což je textový editor (možná varianta k MS Word) a Gnumeric je tabulkový kalkulátor (podobně jako MS Excel). Dokonce je k dispozici i podpora souborových formátů pro MS Office.
Zajímavostí je rozhodně fakt, že veškerý software včetně hotových ukázek a celého obsahu je open source, tedy otevřený, volně šiřitelný zdrojový kód. [36]

3 MATLAB / SIMULINK

3.1 Prostředí Matlab a Simulink

Matlab

Výpočetní a vývojové prostředí Matlab firmy MathWorks je program především pro matematické, technické a vědecké výpočty, grafy, analýzu dat a další související úlohy. Dalo by se to shrnout do kategorie matematika, grafika a programování.
Nabídka je velmi bohatá, vestavěné knihovny dle specifika určení obsahující obrovské množství předchystaných nástrojů. Řeší problémy a zjednodušují výpočty od práce s maticemi až po zpracování signalů, inteligentní energetické sítě, počítačové vidění, bezpečnostní systémy automobilů a strojového učení.

Matlab funguje s vlastním programovacím jazykem a obecně je to nejvýhodnější, ale změna je možná. Výpočtové příkazy lze psát přímo do konzole okna pro okamžitý výsledek (podobně jako klasický kalkulátor), zároveň sestavovat program v okně editoru a potom spustit napsaný kód příkazů buď jako celou sekvenci, nebo po blocích.

Velmi dobře je zpracovaný web MathWorks, který je členěn na sekce komunity se sdílením řešení, problémů, souborů, učení. Sekce Answers, kde jsou videa, diskuzní forum s tříděním na otázky a odpovědi. Vyhledávání se zatržením příslušné oblasti, zda očekáváme odpovědi z podpory, fóra nebo dokumentace.

Obsahuje rozsáhlý seznam popisující jednotlivé funkce a často i vzorové úlohy s již s hotovým řešením včetně popisu (v sekci Examples), většinou existuje k těmto ukázkám odkaz na vyzkoušení ve vlastní instalaci Matlabu zkopírováním a vložením názvu přímo do příkazového řádku. Doplněním je pak klasický Help dostupný offline.

Simulink

Jedná se o nadstavbu Matlabu, která poskytuje grafické zobrazení výpočtových a programových schémat pomocí jednotlivých bloků. Simulink podporuje automatické generování kódu, vestavěné systémy, simulaci dynamických systémů a jejich testování.
Model se vytváří v grafickém rozhraní a tvoří ho prvky spojité a diskrétní, lineární a nelineární, navíc s možností aktivně zasahovat a měnit akční hodnoty v běžící aplikaci.

Knihovna obsahuje bloky pro řešení úloh z různých oborů a dle aplikací od automobilismu - kontrolní systémy; komunikací - procesy zpracování obrazu a videa; elektroniky - Internet of Things, atd. až po finančnictví, biologii, lékařství a letectví.
V knihovně pod složkou Simulink jsou zařazeny běžně používané prvky, což jsou všechny základní pro matematické, logické a signálové operace. Ostatní adresáře mají specializaci na určitou oblast úloh.

Např. pro fyzikální modelování a simulace systémů v elektrotechnice je k dispozici nástroj *Simscape*, který řeší dynamické úlohy z oblasti mechaniky, hydrauliky, elektrotechniky, fyzikálních signálů a termomechaniky. Propojení bloků ve schématu zde simuluje reálné situace, kdy se prvky ovlivňují navzájem a signál probíhá mezi bloky obousměrně.

Součástí blokového schématu jsou procedury a funkce, které běžně obsahují algoritmy psané v Matlabu, ty lze lehce přizpůsobit dle požadavků uživatele.
Hlavní schopností a úkolem tohoto nástroje je přehledné zobrazení a sestavení programu ze základních částí - bloků - jež jsou propojeny signálovými cestami a pracují tak podobným způsobem, jako by byl program napsán ve standardním textovém editoru Matlabu. Knihovnu Simulinku lze doplnit vlastními bloky a při rozrůstajícím se projektu postupně zabalit částí, tzv. subsystémů. I když Simulink jde použít zcela samostatně, u pokročilých projektů je znalost programování v Matlabu nezbytná.

Většinu funkcí a příkazů, které jsou určeny pro prostředí Matlabu, lze použít úplně běžně i v Simulinku. Občas se liší rozdílným názvem, ale odpovídající alternativa je vždy k nalezení. Matlab kodér generuje kód *ANSI C* a podporuje i vytvoření nezávislého spustitelného souboru napsáním vlastní funkce *main.c* nebo *main.cpp* v IDE. [37, 38, 39]

Systémové požadavky

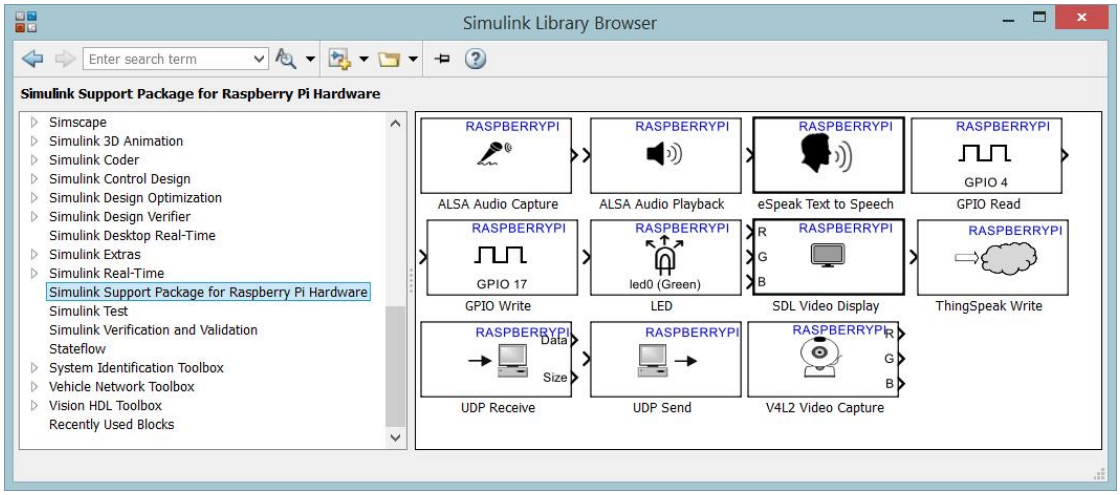
Verze	Procesory	Místo na disku	RAM	Grafika
10 8.1 8 7 SP1	Intel x86-64 AMD x86-64 4 - jádro pro Polyspace*	2 GB typická instalace 4 - 6 GB	2 GB Simulink 4 GB	OpenGL 3.3 1 GB GPU doporučená paměť

Tab 1: Minimální hardwarová konfigurace MATLAB R2016a, R2016b, R2017a pro vybrané operační systémy Windows [37]

* Polyspace je statický nástroj pro analýzu, kontrolu a funkčnost zdrojového kódu [37]

3.2 Simulink Support Package for Raspberry Pi Hardware

Raspberry Pi je jedním z několika produktů kategorie low-cost hardware, pro které lze přidat do Matlabu hardwarovou podporu.

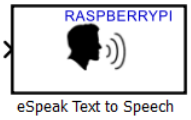


Obr. 6: Knihovna Simulinku a seznam bloků pro Raspberry Pi

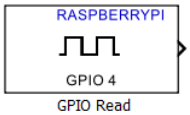
Nainstalováním tohoto balíčku budou programy vytvořené v Matlabu a Simulinku přímo napojeny na komunikaci s daným hw, v našem případě s Raspberry Pi a připraveny na vzájemnou spolupráci.
Podpora komunikace s Raspberry Pi má ve verzi Matlab R2016a k dispozici 11 bloků:

audio vstup, příjem stereo signálu nebo 2 kanálového zvuk. zdroje přes mikrofon (nutná externí zvuk. karta)

audio výstup (nutná externí zvuk. karta). Nejčastěji přes USB, nebo lze použít HDMI

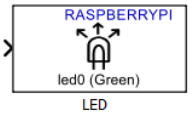


eSpeak funkce – převedení textu na řeč přes hlasový syntetizér

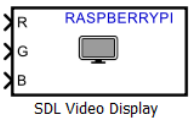


čtení stavových hodnot externího zařízení přes pin porty GPIO (Tabulka s popisem v kap. 2.4)

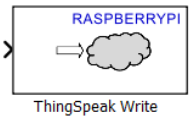
zápis přes pin porty GPIO pro ovládání externího zařízení (Tabulka s popisem v kap. 2.4)



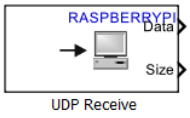
ovládání zelené diody umístěné na desce Raspberry Pi. Slouží hlavně pro otestování a základní seznámení s ovládáním RPi v Simulinku



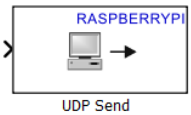
Simple Directmedia Layer (SDL) zobrazuje video data na monitor ve formátu RGB nebo YCbCr.



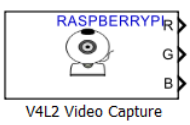
ThingSpeak je webová služba pro aplikace IoT. Umožňuje zpracování a sběr dat v reálném čase, vizualizace, aplikace nebo pluginy



UDP komunikační protokol pro příjem dat. Např pro ovládání zařízení přes mobilní tel.



UDP komunikační protokol pro jednoduché posílání dat. Komunikace RPi s Arduino®, LEGO MINDSTORMS® EV3 a mobilní zařízení



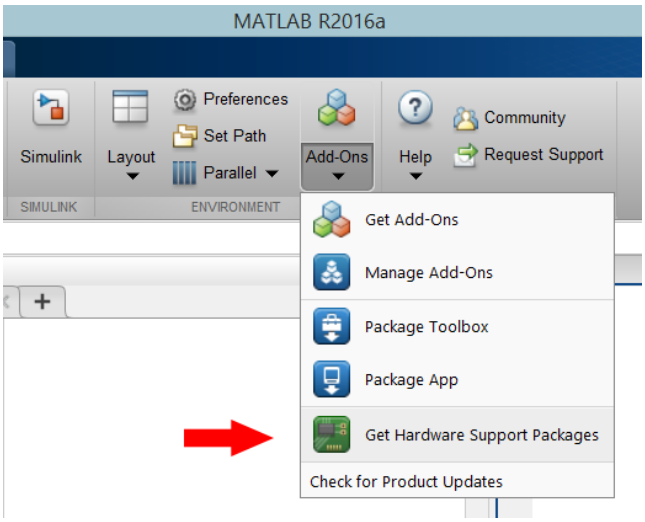
Blok pro příjem obrazových dat z připojené RPi nebo webové kamery. Nastavení video dat ve formátu RGB nebo YCbCr.

Standardní knihovna bloků je k dispozici ihned po vstupu do Simulinku, ale hardwarová podpora pro Raspberry Pi se zde objeví až po dodatečném nainstalování (viz následující kap.). [40, 41]

3.3 Instalace hardwarové podpory pro Raspberry Pi

Co je potřeba mít připraveno

- popis je pro *Matlab R2016a* (EN verzi). Podmínkou je nainstalovaný Matlab verze R2014a nebo vyšší.
- Raspberry Pi s připraveným síťovým adaptérem.
- microSD karta min. kapacita 4 GB, ale lépe pořídit 8 GB, cenový rozdíl je minimální. Je nutno mít adaptér, případně čtečku paměťových karet dle pc, na kterém bude instalace probíhat.
- pro přímé spojení pc s Raspberry Pi kabel RJ45 (LAN) nebo síťovou adresu (IP) pro bezdrátové spojení přes Wi-Fi.



Obr. 7: Spuštění průvodce instalací pro hw podporu

Průběh instalace

Instalace nabízí defaultně adresář předpokládaného umístění Matlabu. Stejné umístění nabídne i instalátor balíčku RPi podpory. Zadat jiný cílový adresář jde pouze předem v *Preferences (HOME > Preferences > Add-Ons)*, protože v průběhu instalování už možnost změny není nabízena.

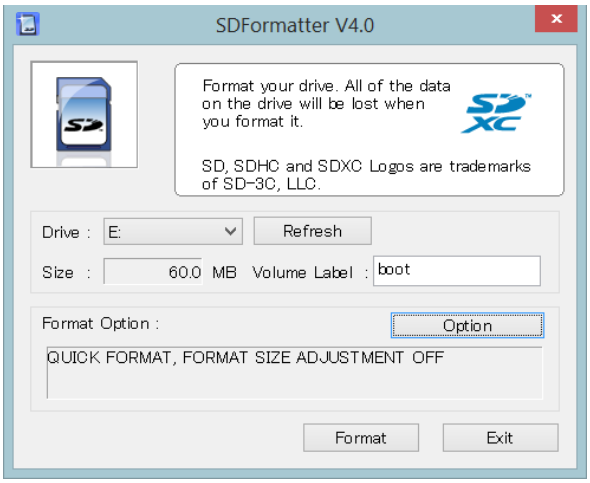
Na kartě *HOME* pod ikonou *Add-Ons* po kliknutí na *Get Hardware Support Packages* se spustí průvodce instalací (viz obr. 7).

Pokud se není účelem odinstalace nebo není instalační soubor stažen předem na lokální disk, je nejběžnější variantou stažení instalačního balíku přes internet, navíc je zaručeno získání poslední aktuální verze.

Po potvrzení se v dalším okně vlevo objeví seznam podporovaných hw, kde stačí označit Raspberry Pi a v pravém formuláři ve sloupci *Action* zatrhnout položku Simulink.

Dalším je „Log In“, kde se požaduje vypsat přihlašovací údaje do účtu MathWorks. Pokud takový účet neexistuje, je možno ho aktuálně vytvořit kliknutím na odkaz vpravo nahoře. Samozřejmostí je zpětný potvrzovací email na zadanou adresu.

Následuje okno s odsouhlasením podmínek. Pak informace o instalovaném produktu, což je operační systém optimalizovaný pro RPi, distribuce *Raspbian Wheezy*.
Po dalším potvrzení už se spouští instalační proces, zde záleží na rychlosti připojení k internetu. Při pomalejším stahování může proces trvat až hodinu. Na konci pak během pár okamžiků problikne automatická registrace hw podpory a aktualizace Simulink knihovny.
Po skončení se objeví okno s informací o úspěšném nainstalování.
Instalační průvodce se pak dotazuje na správný hardware, tedy *Rapberry Pi (MATLAB)* a hned potom i na verzi hw: *Rapberry Pi 3 Model B.*
Pak je tu dotaz na konfiguraci sítě, kde v našem případě bylo zvoleno přímé spojení s hostitelským pc přes kabel ethernet, kvůli nezávislosti místa provozu.



Obr. 8: Program SD Card Formatter [43]

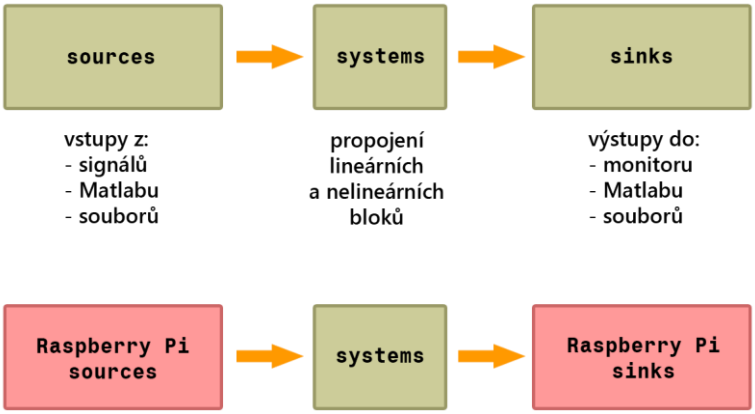
V této chvíli instalátor požaduje vložit paměťovou kartu microSD, kterou sám rozpozná a bude chtít potvrdit. Varování znamená kompletní smazání obsahu na kartě, proto je dobré se ujistit, že se nejedná o jiné připojené zařízení (např. přes USB).

Problém může nastat, pokud má paměťová karta menší kapacitu než je požadovaná. To může nastat z různých důvodů, např. jde o reinstalaci nebo karta už byla použita předtím někde jinde. Provedení klasického přeformátování nepomůže, protože volba ve standardních file managerech jako jsou *File Explorer* (ve Windows) nebo *Total Commander* nenabídne vyšší, tedy původní vysokou kapacitu. Řešením je použití nástroje pro formátování paměťových karet, např. *SD Card Formatter* (viz obr. 8). [43]
Jakmile bude operační systém kompletně na kartě, další instrukce budou už o reálném otestování spojení: microSD zastrčit do slotu na spodní straně Raspberry Pi, propojit ethernetovým kabelem hostitelský počítač s RPi a spustit RPi připojením síťového adaptéru.
Proběhne pokus o načtení, blikne předposlední okno s výpisem IP adresy a tlačítkem pro otestování spojení.
Ukončení s ponecháním zatržené volby pro příklady hw podpory vede do Help prohlížeče Matlabu se vzorovými startovacími příklady. [44]

3.4 Programování Raspberry Pi v Simulinku

Základní struktura

Pro založení nového modelu je třeba spustit Startovací stránku Simulinku buď ikonou Simulinku na domovské kartě *HOME*, nebo přes *New* > *Simulink Model*, případně do příkazového řádku napsat „simulink".



Obr. 9: Typický model Simulinku (nahoře)
a typický model Simulinku pro Raspberry Pi (dole)

Model reprezentuje reálnou fyzickou strukturu. V počítači je to proces dynamického systému složený z proměnných a matematických rovnic, který reprezentují bloky jež jsou funkcí času. Většina těchto procesů je tvořena strukturou typického modelu v Simulinku (viz obr. 9) [47]. Proces simulace může proběhnout oproti reálnému fyzickému zařízení ve zkráceném čase na minimum, což je velká výhoda. Jednou z důležitých věcí je, zda bude model fungovat ve spojitém nebo diskrétním čase a podle toho model sestavovat a volit správné bloky.

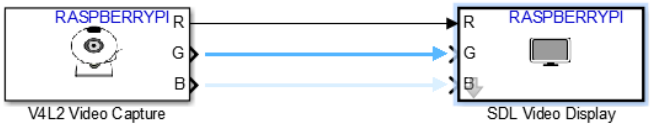
Typický postup tvorby modelu obsahuje:

- Definování systému
- Modelování systému
- Integrace systému

Pro efektivní sestavení modelu musí být jasné základní informace o vstupech a výstupech. Procesní část by měla splňovat požadavky na výstup, jednoduchost a efektivnost z hlediska množství signálových cest a algoritmů. Následuje odstraňování chyb, úprava struktury, sledování odezvy, kontrola hodnot mezivýsledků a výstupu.
Struktura modelu pro Raspberry Pi je typická omezením na seznam bloků určených právě pro vzájemnou komunikaci Simulinku a RPi (viz obr. 9). [45, 46]

Vytvoření modelu

Otevřením startovací stránky Simulinku a kliknutím na obrázek prázdného modelu se objeví editační okno. Dalším krokem je zobrazení knihovny bloků jako zdroje, ze kterého se bude čerpat (čtvercová ikonka v horní liště).

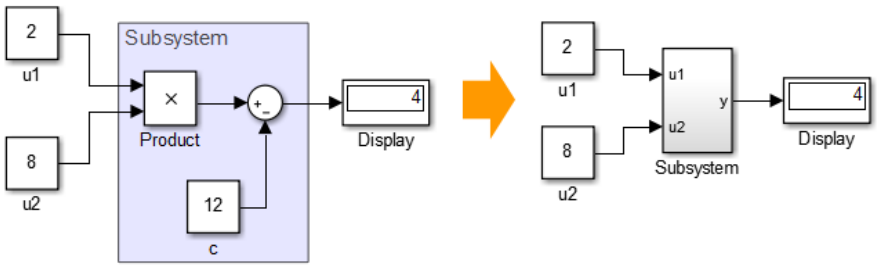


Obr. 10: Bloky a jejich spojení

Pro zobrazení všech dostupných bloků pro Raspberry Pi se označí v levé části knihovny řádek: *Simulink Support Package for Raspberry Pi Hardware* (viz obr. 6). Vybrané bloky je možné přetažením myší přesouvat po jednom do editačního okna. Signálové cesty mezi bloky vzniknou pouze mezi kontakty vstupů a výstupů. Způsobů je více, běžně tažením, nebo podržením *Ctrl* + klik postupně na bloky a nebo kliknutím na počítačem nabízené modře zbarvené automatické naznačení jak je vidět na obr. 10).

Vnořený systém - Subsystem

Jakmile naroste počet bloků a systém začne být nepřehledný, je vhodné zabalit vybranou část bloků do tzv. subsystému, příklad na obr. 11.



Obr. 11: Model s vytvořeným subsystémem

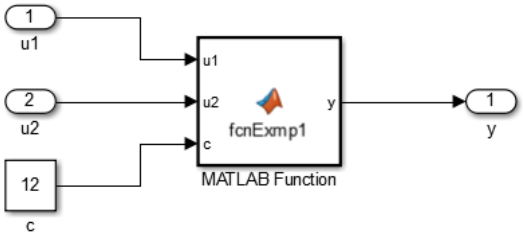
Je to vlastně blok, který funguje podobně jako souborový adresář v počítači. Reprezentuje skupinu bloků, jejichž kontakty pro napojení jsou přivedeny externě. Celá struktura je editovatelná a počet vnořených systémů je libovolný. Pozor na pojmenování vstupů a výstupů (*in/out*) uvnitř, názvy se musí lišit u každého dalšího zanořeného subsystému.

Vlastní definované funkce - User Defined Functions

Často využívanou kategorií ze Simulink knihovny jsou bloky s uživatelsky definovanými funkcemi. Na výběr je více typů, mezi ty zásadní patří:

- *Fcn* je jednoduchý typ se zadáváním funkčního výrazu,
- *Function Caller* je typ volání funkce při vstupní podmínce,
- *Interpreted MATLAB Function* je interpretovaná funkce,
- *S-Function*, načítané funkce z m-souboru, vložení prog. kódu jiného jazyka,

- *MATLAB Function* je pohodlná a hodně populární funkce, individuální nastavení vlastností a parametrů, funguje principiálně jako klasická funkce Matlabu.



Obr. 12: Blok Matlab funkce

Příklad Matlab funkce je vidět na obr. 12. Kód může vypadat např. takto triviálně:

```
function y = fcnExmp1(u1, u2, c)
y = u1 * u2 - c;
```

Poklepáním na blok skočí kurzor do editačního okna Matlabu, kam se kód napíše. Nastaveny jsou 3 vstupy *u1*, *u2* a konstanta *c*, výstup je *y*. Název byl změněn z defaultního *fcn* na *fcnExmp1*. Blok bude fungovat podobně jako systém z obr. 11, kde je ale konstanta schovaná uvnitř subsystému.

Maskování subsystému

Pokud jsou požadavky na určité omezení pro vstupy či výstupy proměnných, je k tomu připraven nástroj maskování. Formulář *Mask Editor* slouží k navolení vlastností portů, parametrů zadávaných proměnných, definování podmínek na viditelnost, přístup, nebo omezení hodnot. Blok pak místo pouhého napojení na vstupy může např. vstupní parametry přijímat v podobě formulářů, nebo z jiného zdroje samostatně.

Při kopírování nebo častější změně parametrů umožňuje maskování editaci hodnot pomocí uživatelsky definovaného formuláře bez zásahu uvnitř subsystému. Slouží také jako určitá ochrana proti nepovoleným změnám, dále je možno použít ikonu na blok, názvy proměnných, přidat popis nebo nápovědu. Pro úlohy v této práci není použit, protože podrobný popis fungování je již v příslušné kapitole a pro názornost je lepší místo skrytí naopak zviditelnění všech proměnných. [45, 46]

3.5 Nastavení modelu Simulinku pro Raspberry Pi

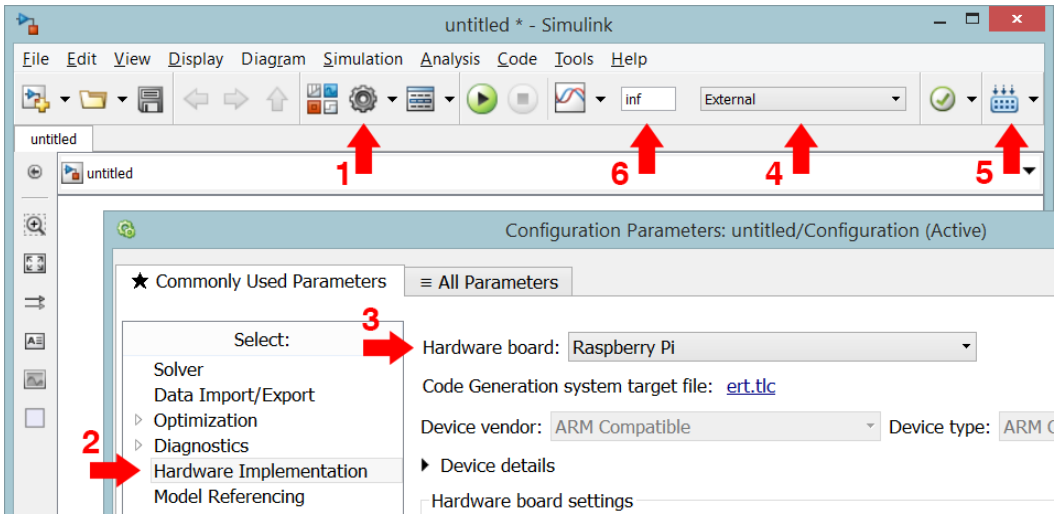
Na hlavním panelu ikon v editačním okně stačí pro vstup do konfigurace jít přes ikonu ozubeného kolečka *Model Configuration Parameters* (obr. 13, č. 1). Zde lze měnit parametry způsobu řešení, času simulace a krokování, konverze dat a typové změny proměnných mezi bloky atd. Pro tuto chvíli ale není třeba do nastavení modelu výrazně zasahovat.

Na kartě *Hardware Implementation* (obr. 13, č. 2) je potřeba první položku *Hardware Board* (obr. 13, č. 3) přepnout na *Raspberry Pi*. Proběhne inicializace a update.

Při nalezení hw budou dole v okně vypsány základní údaje o napojení jako IP adresa a dále přihlašovací jméno s heslem, které musí odpovídat tomu, co bylo nastaveno při instalaci hw podpory.

Problém nastává ve chvíli, kdy po rozbalení hw nabídky tam ona položka *Raspberry Pi* není. Lepší variantou je stav, kdy RPi prostě není zapnuto, nebo v něm není vložena microSD karta s Matlab operačním systémem. Horší je, když řešením je pouze reinstalace celé hw podpory.

Pokud je hw zařízení úspěšně nastaveno na RPi, tak je na hlavním panelu nutno přepnout *Simulation mode* (obr. 13, č. 4) z *Normal* na *External*, což znamená, že simulace má probíhat na externím zařízení, které je nyní nastaveno.



Obr. 13: Editační okno Simulinku a nastavení parametrů pro Raspberry Pi

Tlačítkem *Deploy to Hardware* (obr. 13, č. 5) se program vloží na paměťovou kartu do RPi a bude automaticky spuštěn a probíhat po odpojení od hostitelského pc samostatně, kdykoliv bude Raspberry Pi zapnuto. [48]

Pokud potřebujeme prodloužit simulační čas na nekonečno, vepíše se do formuláře časového ukončení simulace *inf*.

3.6 Raspberry Pi a LED v Matlabu

Zapojení Raspberry Pi s diodou se dá přirovnat ke známému *„Hello world“* příkladu, jakémusi základnímu otestování prvního zprovoznění RPi platformy. Úkolem je nejdříve se v Matlabu připojit k Raspberry Pi a pak napsaným programem rozblikat diodu. Zajímavostí je, že je možné využít diodu na základní desce RPi, takže kromě připojení není nic dalšího potřeba. Pozice diody na desce je vidět na obr. 4 se zeleným popisem *ACT LED*.

Použité zařízení:

- Raspberry Pi 3 B

Program:
Nejdříve se vytvoří objekt *raspi()*, pro manipulaci s objektem bude proměnná *rpi* (příkaz je funkční i bez závorek). Při úspěšném spojení s Raspberry Pi se vypíše tabulka aktuálních stavových hodnot, pokud příkaz nekončí středníkem.

```
rpi = raspi()
```

Zobrazení umístění pinů a diod je příkazem *showPins()*. Jde o informativní obrázek, který je součástí hw RPi podpory Matlabu. Dioda, která má být ovládána, je nahoře vlevo zelená *ACT*.

```
showPins(rpi)
```

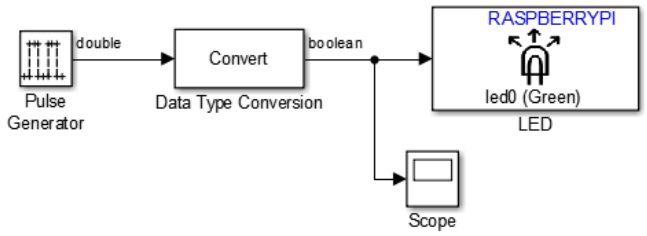
Následuje program. LED bude svítit 1 s, vypnuta bude po dobu 0,5 s. Opakování 3x.

```
for i=1:3
    writeLED(rpi,'led0',1);
    pause(1);
    writeLED(rpi,'led0',false);
    pause(0.5);
end
```

V Matlabu platí, že pro typ boolean hodnotě *true* odpovídá a je zaměnitelné vyjádření číslo 1, pro *false* pak 0. [49]

3.7 Raspberry Pi a LED v Simulinku

Podobně jako v předchozí kapitole jde o stejnou diodu, kód bude nahrazen blokovým schématem, ale místo cyklu *for* bude blok, který umí cyklicky generovat impulsy.



Obr. 14: Schéma zapojení Raspberry Pi a LED v Simulinku [48]

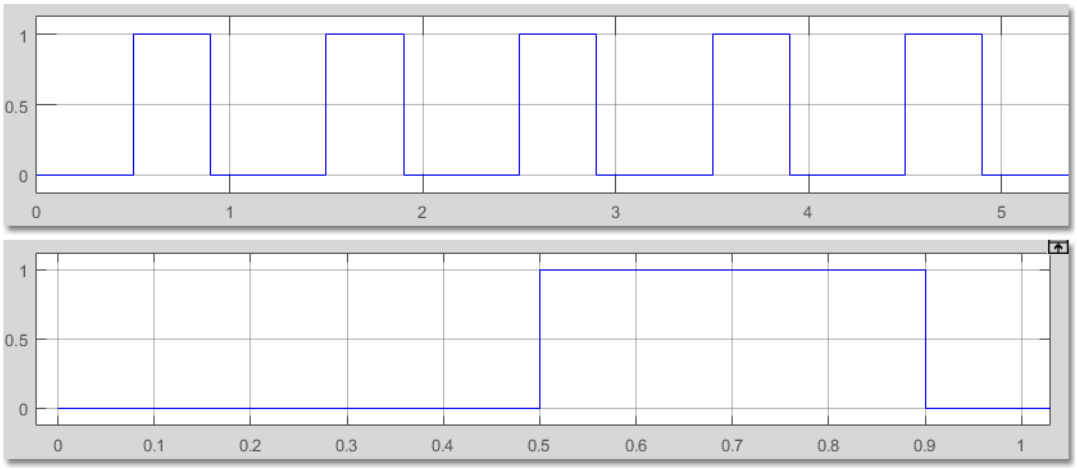
Použité zařízení:

- Raspberry Pi 3 B

Generátor produkuje obdélníkové pulsy s intervalem ve spojitém nebo diskrétním čase dle nastavení, do kterého se vstoupí dvojklikem na blok. Aby byl vidět výsledek na grafu, byl připojen blok *Scope* (viz obr. 14).

Parametry generátoru pro výstup grafu na obr. 15 jsou:

Pulse Type:	Sample Based	Typ impulsů:	diskrétní
Time (t):	Use simulation time	Časování (t):	dle času simulace
Amplitude:	1	Amplituda:	1
Period:	10	Perioda:	10 vzorků (1 s)
Pulse width:	4	Šířka pulsu:	4 vzorky (4/10 s)
Phase delay:	5	Fázové zpoždění:	5 vzorků (5/10 s)
Sample time:	0.1	Čas 1 vzorku:	0,1 s



Obr. 15: Graf pulsů generátoru v bloku Scope, dole zvětšení 1 periody, svislá osa zobrazuje amplitudu a vodorovná čas v sek.

Signál vychází z generátoru s hodnotami pulsů a příchází do bloku LED, který ale funguje na principu vypnout/zapnout a vyžaduje tedy hodnotový typ boolean. Proto je uprostřed vložen blok *Data Type Conversion*, který zajistí správný převod. Ten většinou úspěšně rozpozná vstup a vyhodnotí správný typ proměnné, do něhož signál míří. Stačí ho tedy jen umístit. [48]

Dobrou pomůckou při napojování bloků je určitě zviditelnění výstupních typů proměnných přímo na signální čáře u každého bloku. Zapnutí je v menu na kartě: *Display* > *Signal & Ports* > *Port Data Types*.

4 PŘÍKLAD 1: URČENÍ SMĚRU ZDROJE ZVUKU

Lokalizace zdroje zvuku se objevuje čím dál častěji u počítačů a robotických systémů nejen pro komunikaci s člověkem.

Příkladem jsou systémy, které se orientují v prostoru pomocí příchozího zvuku nebo umí zjistit polohu mluvící osoby, reakce natočení mikrofonu nebo kamery podle často měnícího se aktuálního dění, kvalitnější odposlouchávací zařízení, použití pro kontrolu nebo diagnostiku hlídající změny akustických signálů zařízení v průmyslu.

Zadání

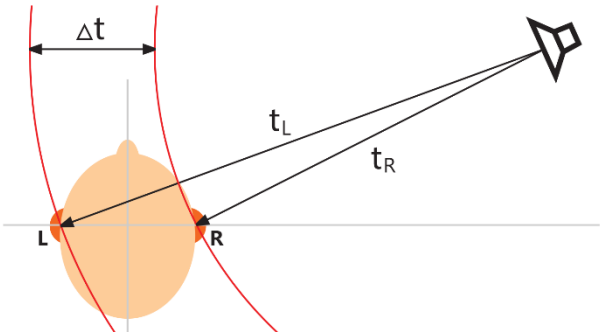
Úkolem je vytvořit model systému, který určí směr příchozího zvuku s použitím stereo mikrofonu.

4.1 Teorie

Problematika zabývající se odhadem směru příchozího vlnění je obecně označována zkratkou *DOA - Direction Of Arrival*. Existují 3 hlavní metody zpracování signálu pro lokalizaci zvuku mikrofonním polem:

- Výpočet podle časového zpoždění *(TDOA)*
- tvarování přijímací charakteristiky *(beamforming)*
- spektrálních odhadech vysokého rozlišení

V tomto příkladu je popisován výpočet podle metody zjištění časového spoždění. Použit je 1 stereofonní mikrofon, což znamená že pro zachycení zvuku je k dispozici 2x mono mikrofon. Pro přesnější lokalizaci je třeba mít snímačů více, nejlépe vytvořit mikrofonní pole. Velký počet snímačů zvyšuje přesnost, ale zároveň i prodlužuje výpočetní čas. [50]



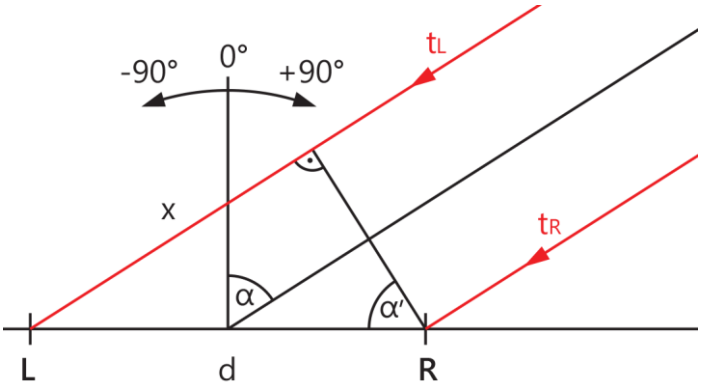
Obr. 16: Rozdílný čas příchozího zvuku (pohled na hlavu shora)

Problematika směru zvuku zahrnuje sice více proměnných pro lidské vnímání, ale z hlediska mechanického příjmu zvuku je hledané řešení pod názvem *Interaural Time Difference (ITD)*, kde se zjišťuje rozdíl ve zpoždění příchozího zvuku a jednou z metod je řešení přes rozdíl fázového posunu. V případě sluchového orgánu jde jednoduše řečeno

o vnímání rozdílu zpoždění poslouchaného zvuku mezi ušima, pokud zvukový signál nepřichází rovnoměrně, ale z jedné strany (viz obr. 16)

Šíření zvukových vln závisí nejvíce na hustotě hmoty prostředí a také teplotě. Obecně se dá říci, že s vyšší hustotou rychlost roste, ale tou zásadní podmínkou je pružnost prostředí. Proto se např. ve vakuu žádné zvukové vlny nešíří a není v kosmickém prostoru nic slyšet. Pro zvukové vlny je lepším prostředím voda než vzduch co se týče rychlosti, dokonce mořská s vyšší hustotou ještě více. Problém je ale směrovost, protože lidský sluch není uzpůsoben na vysoký vnější tlak pod hladinou a mozek nedokáže právě tím porovnáním mezi levým a pravým uchem správně určit rozdíl a vyhodnotit příchozí směr.

Zdroj zvuku pro tento příklad je statický (nepohyblivý) a je zaznamenáván mikrofony rozdělenými na levý (L) a pravý (R) se známou vzájemnou vzdáleností *d* (viz obr. 17) ve slyšitelném rozsahu cca 16 Hz - 20 kHz. [51, 52]



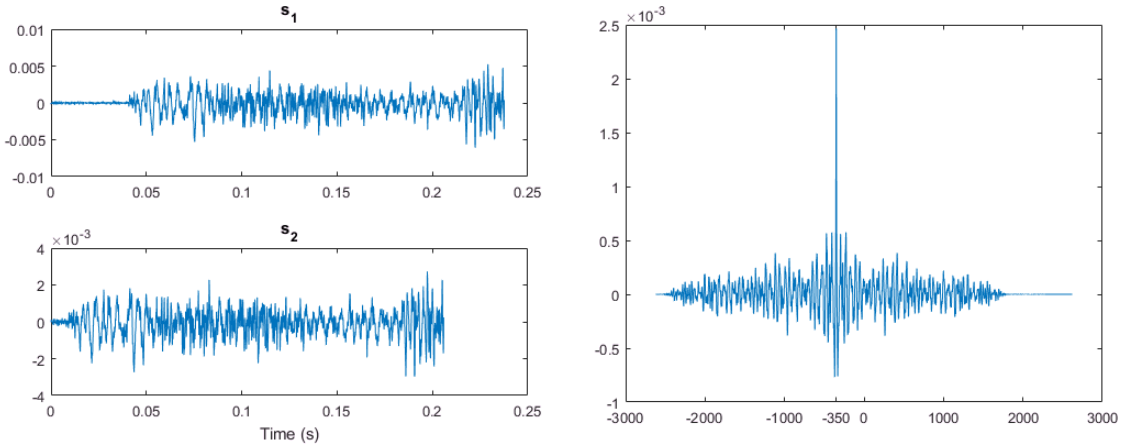
Obr. 17: Rozbor pro výpočet vzdálenosti *x*

4.2 Schéma a popis řešení

Cílem je získaní úhlu α. Pokud by signál šel přesně kolmo na jejich spojnici, znamenalo by to nulový úhel α odklonu od osy, která je označena jako 0°. A když se zdroj zvuku přiblíží spojnici *L* a *R*, bude se naopak úhel blížit ke krajním hodnotám -90° nebo +90°, to znamená maximální časový rozdíl příchozích signálů.

Geometrie je specifikována jako lineární pole mikrofonů v uspořádání (2; 1) v rovině *(ULA – linear uniform array)*. Se zvětšující se vzdáleností klesá přesnost nízkých frekvencí (4 kHz) a pro vyšší frekvence by měly být mikrofony naopak blíže k sobě.
Čas dopadu signálu na levý a pravý mikrofon je označen jako t_L a t_R. Rozdíl těchto časů je zpoždění signálu mezi oběma a je znám jako *Time Delay Of Arrival (TDOA)*. [50]
Vzhledem k tomu, že k dispozici je pouze aktuální průběh dvou stejných signálů, kde jeden bude fázově posunutý, je pro získání této hodnoty vhodnou metodou křížová, neboli vzájemná korelace *(cross-correlation)*. Tato metoda přímo porovnává tvar obou signálů a hledá bod maxima *(peak)* (viz obr. 18). Pro jednoduchost řešení se předpokládá, že zdroj je nekonečně daleko, takže dráhy signálů jsou k oběma mikrofonům rovnoběžné.

Matlab má funkci pro křížovou korelaci *xcorr*, vstupem jsou 2 diskrétní signály a výstupem je zpoždění *(lag)* druhého oproti prvnímu v jednotkách vzorků a vektor hodnot korelace. Jde o nejdůležitější a zásadní výpočtovou část. [53]



Obr. 18: Vlevo nahoře původní signál, vlevo dole signál fázově posunutý a vpravo výsledek výpočtu křížové korelace [53]

Důležitým parametrem je vzorkovací frekvence f_s, která určuje počet vzorků za 1 s. Čas výpočtu jednoho vzorku *samp* (1):

$$samp = \frac{1}{f_s}\ [\text{Hz}] \tag{1}$$

A následně čas zpoždění signálu Δt, ten je roven součinu času trvání jednoho vzorku a počtu vzorků zjištěných jako rozdíl zpoždění vypočtený křížovou korelací (2):

$$\Delta t = samp\ .\ lag\ [\text{s}] \tag{2}$$

Hlavní část výpočtu je hotova. Když je známé Δt, převede se na odpovídající vzdálenost, jakou za tento čas zvuk urazí. Rychlost zvuku vzhledem k nedokonalosti konstrukce aparatury není třeba upřesňovat podle teploty prostředí a stačí běžná hodnota $v = 340\ \text{ms}^{-1}$ [54]. Potom vzdálenost x bude (3):

$$x = \Delta t\ .\ v\ [\text{m}] \tag{3}$$

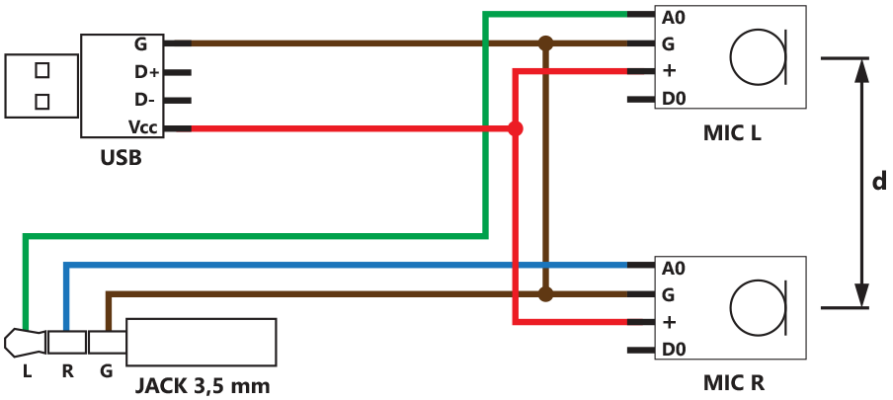
Goniometrickou funkcí z pravoúhlého trojúhelníka se vypočte úhel α', který je díky pravidlu podobnosti úhlů totožný s hledaným α, lze tedy psát (4):

$$\sin\alpha = \frac{x}{d} \quad \rightarrow \quad \alpha = \sin^{-1}\left(\frac{x}{d}\right)\ [°] \tag{4}$$

Matlab počítá úhly v radiánech, takže výsledné hodnoty jsou pak v programu převáděny na stupně. [55]

4.3 Použité zařízení

- Raspberry Pi 3 B
- zvuková karta USB HQ audio MINI adaptér AXAGON ADA-15:
  počet kanálů: 2, (3.5 mm stereo jack),
  vzorkování: 16 / 24 bit,
  vzorkovací frekvence ADC: 44.1 / 48 / 96 kHz,
  odstup signál / šum: (Input SNR) ≥ 90 dB
  pořizovací cena 250 Kč
- mini stereo mikrofon (druhý kontrolní)
  frekvenční rozsah: 100 – 15000 Hz,
  3,5 mm stereo jack, rozměry: 57 x 57 x 15 mm, váha: 9 g
  cenová kategorie do 200 Kč
- stereo mikrofon vlastní konstrukce (viz obr. 19):
  2x mono citlivý mikrofonní modul,
  frekvenční rozsah: 100 – 15000 Hz,
  napájení přes USB (5V),
  3,5 mm stereo jack,
  pořizovací cena součástek cca 150 Kč
  hlavní výhoda: nastavitelná vzájemná vzdálenost mikrofonů



Obr. 19: Schéma zapojení stereo mikrofonu

4.4 Řešení směru zdroje zvuku v Simulinku

Výsledné řešení v příloze 1 - 2.

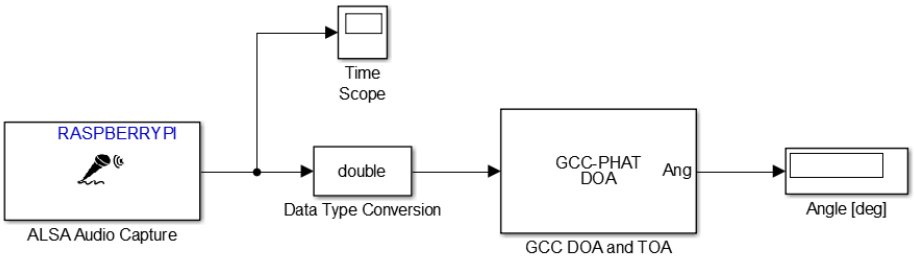
Stereo signál přichází do bloku *LagCalc*, kde je rozdělen na 2x mono, poté je aplikována funkce pro křížovou korelaci *xcorr* a získán rozdíl v počtu vzorků. Ten je pak spolu s dalšími parametry zpracován v bloku *DirCalc*, vypočítána vzdálenost *x* a konečný výsledný úhel převedený z jednotek radiánů na stupně.

4.5 Řešení s využitím Phased Array System Toolbox

V novější verzi Matlab 2016a má Simulink vlastní soubor nástrojů se systémem návrhu a simulace fázového zpracování signálů *Phased Array System Toolbox*. Tento balík nástrojů obsahuje algoritmy pro simulace vysílačů a přijímačů, radarových systémů, analýzu, zkoumání charakteristiky polí senzorů *(beamforming)* a dalších. Jsou zde užitečné funkce a bloky pro automatické řešení zpoždění a určení směru úhlu příchozího zvukového signálu *(DOA).*

Speciálně připravený blok pro Simulink má název *GCC DOA and TOA (Generalized CrossCorrelator with Phase Transform)* a používá kombinaci algoritmů pro obecnou křížovou korelaci s transformací fázového posunu pro určení příchozího směru signálu *(GCC-PHAT DOA).* Kromě základního úhlu *(Ang)* je možnost přidat další dva výstupy, hodnotu křížové korelace a čas zpoždění *(TOA).* [56]

Výsledné řešení:



Obr. 20: Schéma s předdefinovaným blokem *GCC-PHAT DOA*

Omezení Phased Array System Toolbox

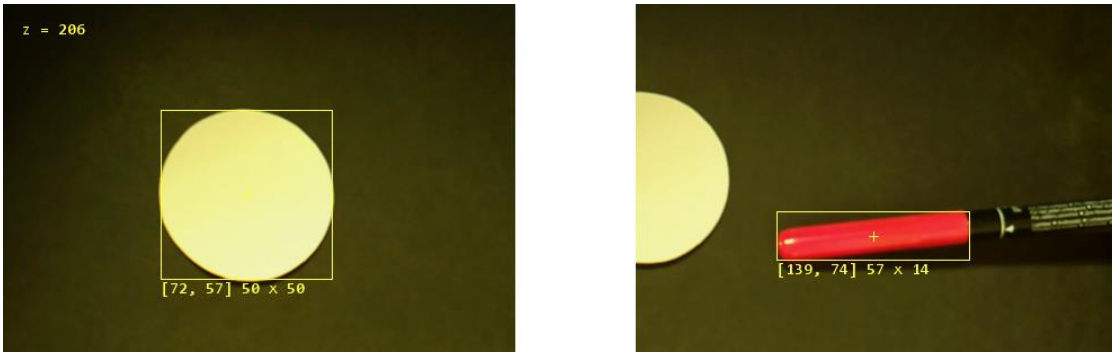
Objektové metody, které nejsou podporovány: *patternAzimuth, patternElevation, plot, plotResponse a viewArray.*

Není podporováno generování kódu pole, který obsahuje antény nástroje *Antenna Toolbox.*

Při použití antén a polí, které vytvářejí polarizovaná pole, musí být parametr *EnablePolarization* pro tyto objekty systému nastaven na hodnotu *true.* [56]

5 PŘÍKLAD 2: SLEDOVÁNÍ OBJEKTU

Sledování a detekce přítomnosti objektu je úzce spojena s problematikou počítačového vidění a technického zpracování obrazu a videa. Jde o stále více a rychleji se rozvíjející oblast širokého použití hlavně v automatizaci, průmyslu, automobilismu, letectví, nebo v armádě. Po těchto systémech se požaduje, aby uměly samostatně zpracovávat obraz a získávat z něj informace potřebné pro svou činnost. Jsou součástí řídících systémů jako autonomní řízení aut a navigace obecně, řízení robotů, analýza lékařských snímků, biometrické údaje, defektoskopie, střežení objektů a dalších.Výhody jsou rychlost zpracování, množství, paralelnost při současném zkoumání více parametrů, oproti lidskému faktoru rozhodně neunavitelnost, nezávislost a relativní bezchybnost.



Obr. 21: Záběry z kamery: průběh kalibrace (vlevo) a sledování objektu (vpravo)

Zadání

Úkolem v tomto demonstračním příkladu je vytvořit program, který bude schopen detekovat objekt na principu segmentace barvy a sledovat ho při pohybu. Součástí je kalibrace, která vypočítá reálné souřadnice a rozměry detekovaného objektu. Kalibrační proces zároveň zjistí vzdálenost kamery a pozadí pomocí kalibračního tělesa.

5.1 Teorie

Metod zpracování obrazu je celá řada a vždy je nutno individuálně přizpůsobit řešení danému účelu.

Barevné modely

- **RGB** - červená *(Red)*, zelená *(Green)* a modrá *(Blue)* značí aditivní model, složen ze 3 základních složek barvy. Sytost složek a jejich kombinace pak vyjadřuje ostatní barvy spektra. Používá se tam, kde se barva zobrazuje pomocí světla, nejlépe neosvětlené pozadí jako jsou monitory (LCD) nebo promítací zařízení.
- **CMYK** - jedná se substraktivní bar. model, složky barev jsou azurová *(Cyan)*, purpurová *(Magenta)*, žlutá *(Yellow)* a navíc přidaná černá *(Key black)*. Na rozdíl od RGB modelu bývá světlé pozadí (nejlépe bílé), na které jsou barvy tisknuty - přidáváním barevných složek výsledná barva tmavne a barevnost naopak ubývá.

Kompletní barevnou škálu z RGB nelze přesně zobrazit CMYK modelem a naopak. S tím se pojí známé problémy tisku.

- **HSV** - tón *(Hue),* sytost *(Saturation)* a jas *(Value).* Tón znamená změnu odstínu barvy, Sytost se mění v intervalu <0;1> a jas pohybuje výsledkem od nejtmavší, hodnoty (černé) po nejsvětlejší sytost daného tónu barvy.
- **YUV** - luminance, jas *(Y),* UV chrominance *(UV).* tento model byl vytvořen pro televizní vysílání a video, pro kompatibilitu s černobílým vysíláním. Přidáním barevné jasové složky se příliš nezvětšila šířka přenosového pásma.
- **YCbCr** - luminance, jas *(Y),* modrá chrominance *(Cb)*, červená chrominance *(Cr).* Informaci o barvě nesou pouze tyto 2 chrom. složky. Použití pro dig. fotografie a video z důvodu lepší komprese. Bývá často převáděn z prostoru RGB přesně definovaným typem matice. [57]

V případě úlohy pro Raspberry Pi je důležitým hardwarem připojená kamerka, která snímá scénu a přenáší obraz videa ke zpracování do modelu Simulinku. Ten nabízí na výstupu z bloku kamery 2 barevné modely, RGB a YCbCr.

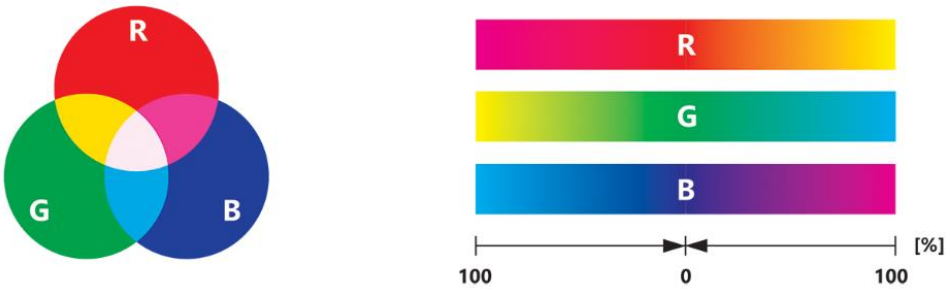
Pro pokročilejší způsoby segmentace objektu, jako jsou úlohy na rozpoznávání objektu nebo např. detektor lidí, jsou v Matlabu vytvořeny už hotové procedury a tam je YCbCr barevný prostor výhodný. Pojí se s tím ale některé problémy. Pro rozpoznávání objektu ve videu je třeba definovat hledaný předmět, který musí být uložen v mnoha variantách zobrazení, pro spolehlivou funkčnost řádově v tisících. Slouží k tomu utilita integrovaná v Matlabu, označena jako trenažér: *Training Image Labeler* a lze ji aplikovat pouze na videozáznam. Podobně jiné jako *peopleDetectorACF* nebo výborná metoda pro rozpoznávání textu *OCR (optical character recognition)*, bohužel funkční jen pro obrázky, ne videa. Balík nástrojů *Simulinku Computer Vision System Toolbox* obsahuje bloky pro pokročilé zpracování obrazu, ale vzhledem k typu úlohy, kdy objekt bude sledován kamerou naživo a bude se pohybovat v prostoru s nehomogenním pozadím, je vhodnou metodou segmentace objektu na základě barvy použitím modelu RGB. Podmínkou je, aby objekt obsahoval jednu výraznou převládající barvu nebo aspoň dostatečně velkou plochu této barvy odlišné od okolí. [58]

Detekce objektu musí mít určeno jisté toleranční pásmo, do jaké míry může přesahovat barva do sousedního spektra.

Reálné souřadnice objektu a vzdálenost kamery od pozadí (*z*-souřadnice) řeší kalibrační proces s pomocí kalibračního tělesa. Ten musí proběhnout jen v určitý okamžik živého přenosu videa při neustále měnících se podmínkách, navíc jeho platnost bude jen po dobu nezměněné *z*-souřadnice. V opačném případě přestanou zkalibrovaná čísla platit a je třeba kalibraci opakovat. Průběžná kalibrace by byla možná, pokud by kalibrační těleso bylo neustále přítomno v záběru kamery a přitom nenastalo překrytí sledovaným objektem. Problematikou výpočtu vzdálenosti se zabývá např. stereoskopie.

5.2 Schéma a popis řešení

Na obrázku 21. jsou vpravo zobrazeny přechody, stupnice pod nimi znázorňuje množství přimíchané sousední barvy v procentech, přičemž každý panel obsahuje 100 % hodnoty své hlavní barvy (100 % odpovídá číslu 255 při 8-bitovém rozlišení).

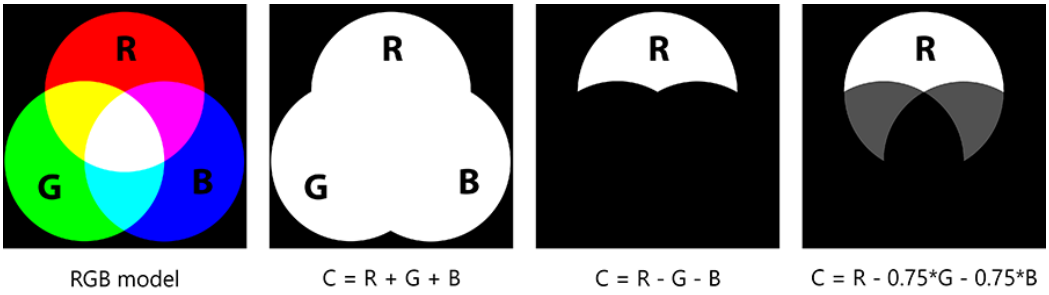


Obr. 22: RGB model (vlevo) a jejich vzájemné přechody (vpravo)

Př.: Nechť hlavní detekovaná barva je *R* a toleranční pole s přimícháním sousedních barev (*G, B*) 25 %, pak obecná rovnice výsledné barvy ozn. *C* bude (5):

$$C = R - 0{,}75*G - 0{,}75*B \qquad (5)$$

Výsledkem barvy C jsou body detekovaného objektu. Barevné znázornění je na obr. 23.



Obr. 23: RGB model a selekce červené barevné složky

Použit je RGB model z obr. 22, zleva doprava: 1.) plné RGB zobrazení, 2.) (*BW)* všechny barvy, 3.) izolovaná pouze červená složka, 4.) vybrána červená barva s prolnutím 1/4 ostatních složek. Regulování množství obsažené barvy je většinou o kompromisu, kdy jde o posouvání a hledání ideálního nastavení hranice intenzity (*threshold*) vybrané barvy. S větší hodnotou se sice zvyšuje citlivost a schopnost zachytit více odchylek barvy objektu, ale na druhou stranu se zvyšuje riziko rozšíření na nežádoucí okolí nebo jiné předměty. Závisí to na více faktorech, jako je typ a barevné složení detekovaného předmětu, na aktuálním osvětlení apod.

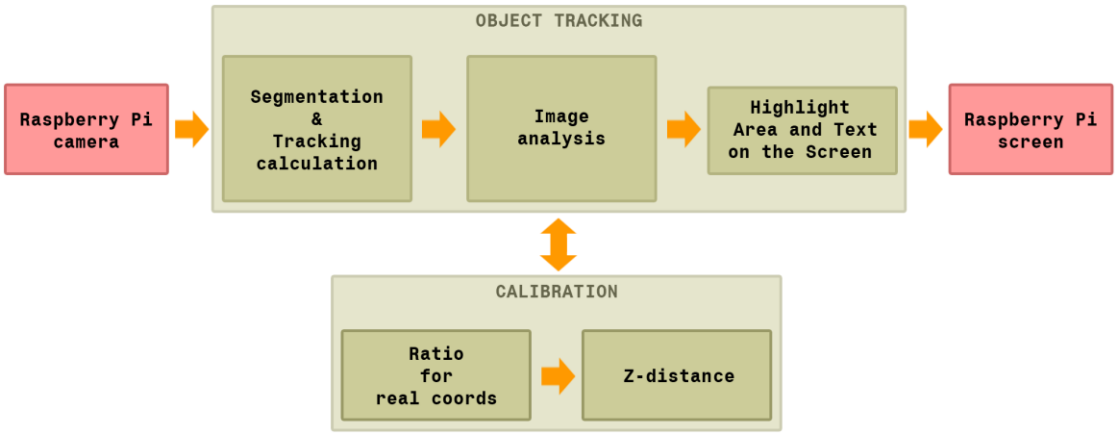
Simulink má předdefinovaný blok pro statistickou analýzu oblasti, kde vstupem je binární obraz *BW*, který vrací mimo jiné obdélníkové souřadnice této oblasti a velikost plochy. Jakmile je oblast definována, je třeba určit a správně přepočítat souřadnice (viz. kap. Kalibrace). Posledním článkem procesu je zobrazení, kde je nutno přiložit

k původnímu obrazu všechny zvýrazněné údaje o sledovaném objektu a zobrazit je na monitor.

Základem jsou vstupní konstanty, které jsou uživatelsky měnitelné i za běhu programu:

- typ detekované barvy (RGBW), lze zvolit 1 ze 4 složek spektra
- hranice intenzity detekované barvy (*threshold*), hodnoty 0 až 255

Souřadnicový systém je definován tak, že počátek leží v levém horním rohu, kladný směr osy *x* jde doprava a osa *y* dolů (viz obr. 25). Proces sledování objektu se skládá ze 2 hlavních programových částí, které navzájem spolupracují (schéma diagramu viz obr. 24:



Obr. 24: Diagram zpracování procesu sledování objektu a kalibrace

Sledování objektu:

- Vstup z kamery připojené na Raspberry Pi
- Detekce objektu na základě segmentace barvy
- Analýza velikosti plochy a souřadnic obdelníku obsahujícího objekt

Kalibrace:

- Zjištění poměru zobrazení pro reálné souřadnice a rozměry
- Výpočet vzdálenosti kamery a pozadí jako z-souřadnice

5.3 Použité zařízení

- Raspberry Pi 3 B
- Kamera Pi NoIR v1, (NoIR = No Infrared) bez IR filtru, upevněno na ministativ, použitelné max. rozlišení 640 x 480 px (30 fps),
  zorný úhel 54° x 41° [60]
- Testovací pozadí: Černá papírová plocha
- Kalibrační těleso: vystřižený kruh z bílého papíru, průměr 50 mm

5.4 Řešení sledování objektu v Simulinku

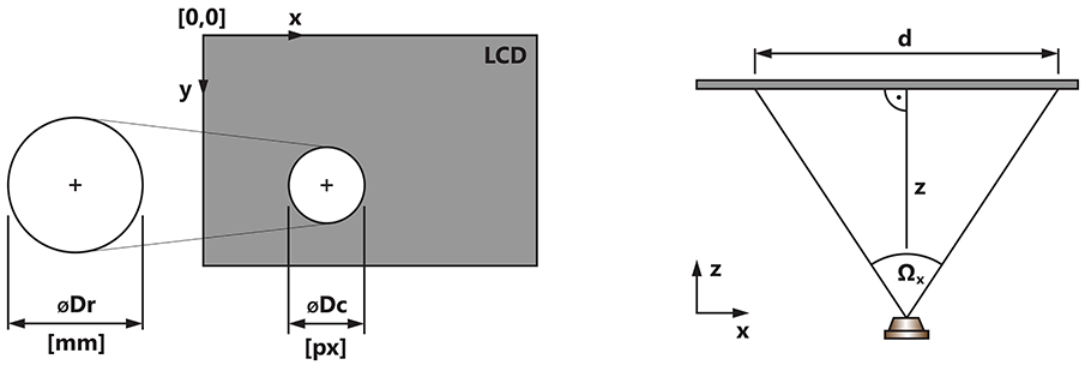
Výsledné řešení v příloze 3 - 6.

Kamera RPi (*V4L2 Video Capture*) snímá obraz ve složkách RGB s max. použitelným rozlišením 640 x 480 px a přichází do bloku *TCalc*, kde je detekována barva objektu. Tato oblast je převedena na binární obraz a vstupuje do bloku *Blob Analysis*, kde je zjištěn celkový obsah obrazu (*Area*), souřadnice obdélníku (*Bbox*) zahrnující tento obraz a výsledný počet těchto oblastí (*Count*). Ten byl stanoven na jednu, která je použita jako informace o existenci detekce.

Následně jdou proměnné do subsystému *Highlight Area and Text on the Screen*, což je procedura pro zobrazení informací na monitoru. Pokud není detekován objekt, na monitor půjde pouze text, že nebylo nic zjištěno (*Insert Text NOT DETECTED*). V opačném případě blok pro výpočet souřadnic (*ViewCoords*) převezme rozměry detekované oblasti a upraví je na souřadnice, na které aplikuje vypočtený poměr zobrazení z výsledku kalibračního procesu a převede je na reálné jednotky v *mm*. Signál RGB projde s postupným přidáním zmíněného obdélníku, poté přidáním středu a nakonec textem již reálných souřadnic vypsaných pod ozn. obdélníkem. V hranaté závorce jsou souřadnice, vpravo šířka a výška oblasti, vše v *mm*. Na výstup jde celý výsledek z RPi bloku (*SDL Video Display)* na monitor.

5.5 Kalibrace

Kalibrace slouží ke zjištění poměru mezi zobrazovanými a reálnými rozměry s převodem obrazových bodů (*px*) na reálné jednotky (*mm*).



Obr. 25: Kalibrační těleso reálné a zobrazené (vlevo), kamera a pozadí pro výpočet vzdálenosti *z* (vpravo)

Po provedené kalibraci se informace o souřadnicích sledovaného objektu, jeho rozměrech a *z*-souřadnici (vzdálenost od kamery) zobrazují na monitoru v *mm*. Výsledek je vidět na záběrech kamery na obr. 21, kde vlevo je dokončený úspěšný proces kalibrace a vpravo již aktuálně měřený sledovaný objekt.

Souřadnice v hranaté závorce určují polohu (viz. souř. sys. obr. 25) a další čísla vpravo znamenají velikost detekované oblasti. Pro použití kalibrace se předpokládá tmavé až černé pozadí - podložka. Kamera musí mířit kolmo na ni (viz obr. 25 vpravo). Aby mohl být poměr vzdálenosti kamery a pozadí vypočítán, je nutné znát několik parametrů (jsou uživatelsky nastavitelné):

- průměr reálného objektu pro kalibraci (*Diameter of real object*)
- rozlišení videa v ose x (*Video resolution x*)
- zorný úhel kamery v ose x (*Angle of camera lens in x*)

Kalibrační těleso

Nejvhodnějším tvarem je kruh bílé barvy a minimální hloubky, např. vyrobený z papíru. U kruhu nezáleží na pootočení a bílá barva je nejjasnější pro detekci, v modelu ozn. *W*. Přesto vzniká při měření velikosti objektů jedna zásadní nepřesnost, protože každé těleso má určitou hloubku a také je nějak vzdáleno od změřeného pozadí. Praktické využití proto bude mít hlavně měření vzdálenosti kamery od pozadí. Např. pro staticky umístěnou kameru s kolmo stavitelným pozadím. Využití se může objevit třeba jako součást kalibrace 3D tiskáren, v měřících bezkontakních zařízeních v průmyslu jako optický doplněk laserových měřičů atd.

Výpočet poměru zobrazení

Kalibračním tělesem je kruh o známém průměru D_r v *mm*. Průměr tohoto kruhu na monitoru je D_c v *px* (viz obr. 25 vlevo). Výpočet přes obsah *S* (6) je přesnější než přes průměr, navíc obsah zobrazovaného tělesa velmi přesně spočítá blok *Blob Analysis*.

$$S = \frac{\pi D_c^2}{4} \quad \rightarrow \quad D_c = \sqrt{\frac{4S}{\pi}} \quad [\text{px}] \tag{6}$$

Poměr zobrazení *Ratio* mezi skutečným (D_r) a zobrazeným (D_c) průměrem je (7):

$$Ratio = \frac{D_r}{D_c} \tag{7}$$

Výpočet *z*-souřadnice

Pokud je známa hodnota *Ratio* a rozlišení obrazu videa v ose *x* je *d*, zorný úhel kamery v ose *x* je Ω_x (viz obr. 25 vpravo), pak vzdálenost *z* získaná goniometrickou funkcí bude:

$$\tan\left(\frac{\Omega_x}{2}\right) = \frac{\frac{d}{2}}{z} \quad \rightarrow \quad z = \frac{d}{2\tan\left(\frac{\Omega_x}{2}\right)} \cdot Ratio \quad [\text{mm}] \tag{8}$$

5.6 Řešení kalibrace v Simulinku

Výsledné řešení v příloze 7 - 8.

Poměr zobrazení (*Ratio*)

Známý obsah plochy reálného tělesa tvaru kruhu v *mm* (*Dr*) je zadanou vstupní konstantou. Pokud je kalibrační těleso bílé barvy, je potřeba otočit přepínač barvy na bílou (*W*) a jakmile je oblast celá detekována, spustit kalibrační process tlačítkem *CALIBRATE*.

Jednoduchou kontrolou lze zjistit, zda byl process úspěšný. Rozměry objektu na monitoru se musí shodovat se skutečnými. Pokud se objeví nepřesnosti, posuvníkem citlivosti detekce barvy (*Threshold*) se detekovaná oblast upraví a proces stačí zopakovat.

Z - souřadnice

Součástí procesu kalibrace je zároveň výpočet vzdálenosti kamery od měřeného pozadí, na kterém je umístěno kalibrační těleso. Vzdálenost (*z*) je výpočtem ze známých zadaných konstant rozlišení kamery a zorného úhlu kamery upravena poměrem zobrazení (*Ratio*). Výsledkem je hodnota v levém horním rohu monitoru v *mm*.

6 PŘÍKLAD 3: 3D POLOHOVÁNÍ KAMERY

Tento příklad navazuje s malými změnami na předchozí úlohu, kde sledování objektu bylo popsáno z hlediska staticky umístěné kamery.

Pokročilé sledování objektu se neobejde bez fyzického pohybu kamery, pokud se neberou v úvahu speciální případy jako optika s tzv. 360° zorným úhlem apod. Použití a aplikace v oblasti automatického monitoringu, kontrole ve výrobě, CCTV a místa ostrahy nebo nestandardní chování objektu, doplněk radarových systémů a další případy, kde je zájem o sledování nebo udržení zaměřeného objektu na kameře. Pro účinné sledování v prostoru je pohyb kamery min. ve 2 osách podmínkou.

Obr. 26: Polohovací konstrukce pro kamerku Raspberry Pi a záběr z kamery

Zadání

Vytvořit pohyblivé 2-osé rameno a ovládací program pro Raspberry Pi kameru, která se bude automaticky otáčet a sledovat detekovaný pohybující se objekt.

6.1 Teorie

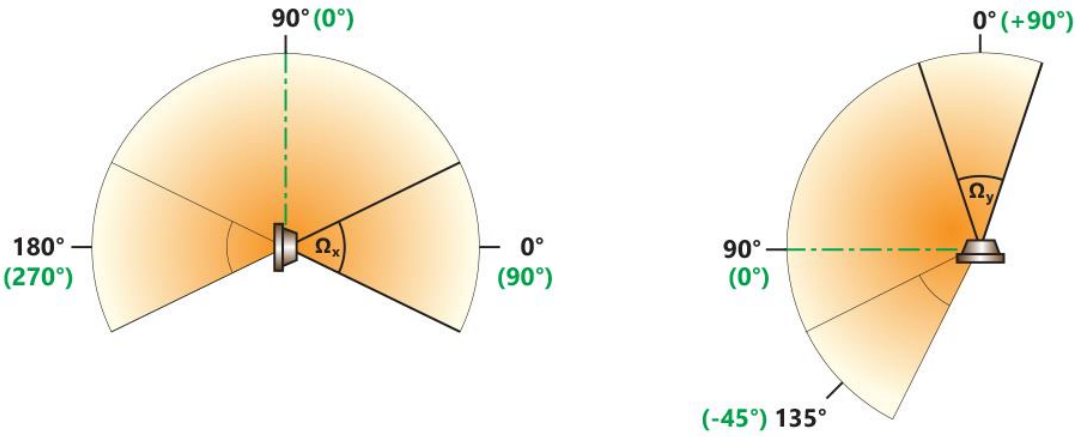
Pro řešení této úlohy je potřeba napojit mikroservomotory na Raspberry Pi. V tomto případě použitý počet 2 kusů je také maximální počet, který jde zprovoznit přes GPIO při přímém spojení bez dalšího pomocného příslušenství. RPi disponuje pouze dvěma piny napájení 5V (mikroservo začíná na cca 4,8V), navíc je potřeba počítat s vyšším napětím adaptéru do RPi, který musí utáhnout kamerku a případně další spotřebiče. Pro případné zapojení většího počtu zařízení by bylo nutné pořídit některý vhodný shield s externím napájením, pak už jde jen o omezení počtu datových pinů. Při návrhu polohovací konstrukce se musí zohlednit krátká délka datového kabelu ke kameře, aby hlavně výška v nejvzdálenější poloze kamery dostačovala.

Zásadní zádrhel ale nastává při pokusu zprovoznit serva v prostředí Simulinku, protože hw podpora pro RPi neobsahuje bloky pro jejich ovládání. MathWorks uvádí bohužel jen příklady pro jiné platformy, hlavně pro Arduino, což je mikrokontrolér. Pro Matlab je sice popsáno ovládání serva, ale příkaz je doprovázen chybou o neznámé

funkci. Vzhledem k povaze demonstrační úlohy, ve které nejde o přesné statické polohování, ale o relativní, není ani nutná synchronizace nebo krátká reakční doba, je možné Raspberry Pi použít s tím, že toto omezení se obejde vytvořením vlastního algoritmu. Toto řešení ale slouží jen jako prezentace schopností Simulinku, protože plnohodnotné ovládání serva by bylo lepší pomocí funkce implementované v Matlabu a přímo přizpůsobené podmínkám provozu RPi jako je např. pro Arduino (i když ani tam to není úplně ideální) nebo použít některý *shield* (HAT) modul pro generování PWM signálu. Signál přicházející do RPi přes LAN z jiného pc vykazuje nepřesnosti v chování serva při polohování a také překmity. [59, 62]

6.2 Schéma a popis řešení

Kamera se bude otáčet v horizontálním a vertikálním směru, pohyb zajistí dvě mikroserva. Startovní poloha kamery je nastavena na ideální střed tak, aby byla v polovině maximálního rozsahu serva v obou rovinách (obr. 27, přeruš. zelená čára).



Obr. 27: Rozsah pokrytí kamery v horizontální (vlevo) a vertikální rovině (vpravo), v závorce sférické souřadnice, vlevo azimut, vpravo elevace

V horizontální rovině má kamera zorný úhel Ω_x = 54° a využije tedy celých 234°. Ve vertikálním směru dosáhne kolmo nahoru až na +90°, ale směrem dolů od horizontu je omezena z důvodu stínící konstrukce ramene na -45°, takže při zorném úhlu kamery Ω_y = 41° ve vertikální rovině má efektivní působení 176°, viz. obr. 27. Ve chvíli, kdy objekt unikne ze záběru, se kamera vrátí zpět na startovní polohu (ozn. zelenou přeruš. čarou), kde počká na nový objekt.

Polohovací konstrukce

Cílem je udržení celé detekované oblasti v zorném poli kamery, konstrukce ramena tedy může vykazovat určité nedostatky, aniž by to zásadně ovlivnilo výsledek. Např. není třeba uvažovat chybu paralaxe. Konstrukce je primitivní, částečně rozkládací. Skládá se z trojnožky ministativu jako stojanu, 2 podobných L profilů pružného plastu a na každém z nich připevněno jedno servo. Většina spojů je slepených na pevno, kamera je připínací

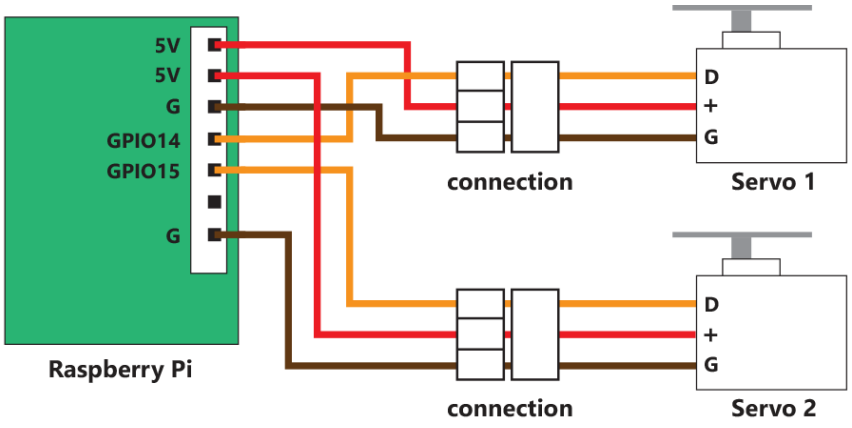
tvarovaným drátkem. Celá konstrukce je přesto rozkládací na 3 hlavní části, stojan s horizontálním ramenem, vertikální rameno a kamera (viz obr. 26).

Sférické souřadnice

Počátek souřadnic se oproti předchozímu příkladu přesune na střed optiky kamery, a protože již nejde o měření objektu v Kartézském souřadnicovém systému, ale na sférické ploše, zobrazují se pouze úhly polohy, systém je naznačen na obr. 27 v závorce. Pro horizontální rovinu to je azimut, ozn. *A*, a pro vertikální rovinu elevace ozn. *E*. Vzdálenost je neznámá a nelze ji s použitým vybavením určit, nemá ale žádný vliv na ostatní výpočty. Hodnoty úhlů mají nulový bod ve startovní pozici kamery a jsou počítány s ohledem na její pohyb v obou směrech. Souřadnice musí ukazovat vždy správné hodnoty nezávisle na poloze objektu v ploše obrazovky (1. fáze výpočtu) a na poloze natočení kamery (2. fáze výpočtu). V 1. fázi proběhne analýza pozice objektu, v jakém kvadrantu se nachází a odvození lokální pozice na obrazovce. Ve 2. fázi pak přepočet úhlů natočení servomotorků na úhly ve smyslu shodné orientace se sférickým souřadnicovým systémem 1. fáze (viz obr. 27, převod z „černých" čísel na „zelená"). Výsledky úhlů *A* a *E* jsou pak kombinací obou výpočtů a jsou vypisovány pod detekovanou oblast (viz obr. 26 vpravo).

6.3 Použité zařízení

- Raspberry Pi 3 B (+ silnější 3A adaptér napájení)
- Kamera Pi NoIR v1,
  zorný úhel 54° x 41° [60]
- 2x mikroservo G0-09 (analog, používané pro RC modely)
  napájení 4,8 - 6V, kroutící moment 1,3 kg/cm
  rychlost: 0.12 s / 60° (4.8V)
  schéma zapojení obou servomotorků do RPI je na obr. 28
- Polohovací konstrukce vlastní výroby pro umístění kamerky a serv (obr. 26)



Obr. 28: Schéma zapojení obou mikroserv na GPIO Raspberry Pi

6.4 Řešení ovládání serva v Simulinku

Výsledné řešení v příloze 11.

Základní součástí řízení servomotoru je pulsně šířková modulace (PWM), Simulink schéma je dokonce k nalezení na webu podpory MathWorks [61]. Přestože toto bylo úspěšně otestováno, tak byl pro zjednodušení vybrán z knihovny hotový blok na ovládání DC motoru *PWM Generator (DC-DC)*, kde vstupem je šířka pulsu v procentech. Úhel natočení záleží pouze na šířce pulzů, ne na opakovací frekvenci. Nejběžnější hodnoty pro standardní ovládání serv jsou: frekvence 50 Hz, tedy časová perioda (rámec) 20 ms, šířka pulsů, která je úměrná výchylce otočení serva je typicky 1 až 2 ms, obecně viz obr. 29. Procentuálně to vychází na 5 % až 10 %, ovšem výjimek je hodně a mikro servo použité v tomto projektu je jednou z nich. Experimentálně bylo naměřeno účinné rozmezí šířky pulsů (tab. 2):

Úhel natočení [°]	Šířka pulsu [s]
0	0,0005
180	0,0025

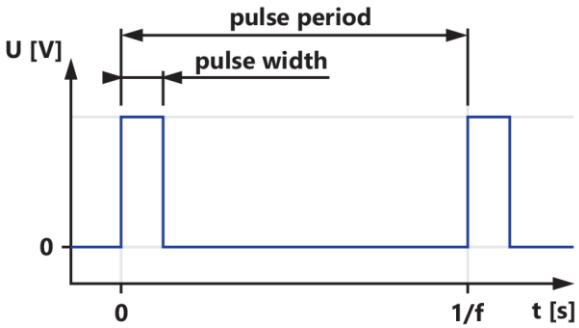
Tab. 2: Naměřené hodnoty rozsahu serva

Tím, že blok *PWM Generator (DC-DC)* má vstup v procentech, vychází vstupní hodnoty na obě krajní polohy vzhledem k periodě (tab. 3)::

Šířka pulsu [%]	Šířka pulsu [s]
100	0,0200
2,5	0,0005
12,5	0,0025

Tab. 3: Charakteristika pulsních hodnot serva

Výsledkem je interval délky pulsů v procentech a do bloku generátoru přijdou tedy hodnoty (0,025; 0,125). [62]



Obr. 29: Pulsně šířková modulace (PWM) pro ovládání serva

Převedení rozsahu pulsů serva na stupně otočení pomocí interpolace
(Řešení v příloze 14.)
Efektivní hodnoty pro mikroservo byly nastaveny na délku pulsů (0,025; 0,125) při periodě 20 ms (50 Hz) a při 5 V. Tento interval je pro lepší funkčnost a doraz na krajní polohy rozšířen na (0,02; 0,13).

Pomocí jednoduché interpolace (příloha 14) jsou čísla převedena ze vstupu ve stupních otočení na procenta šířky pulsů pro blok *PWM Generator (DC-DC)*. Na modelu si lze všimnout zvláštnosti, že blok *Substract* je použit jednou pro odečítání a podruhé pro součet, podobně blok *Divide*. V knihovně jsou tyto bloky pod správnými názvy *Product* (součin) a *Add* (součet). Jedna z výhod Simulinku je právě editovatelnost parametrů, proto může být stejný blok na více místech s rozdílnou funkcí. Samozřejmě je pro lepší přehled používat správné názvy, nebo je následně přejmenovat.

Problém překmitů
(Řešení v příloze 15.)
Ve chvíli, kdy má být servo bez vnějšího zásahu v klidu, neustále kmitá kolem své polohy. Problém těchto rušivých překmitů byl vyřešen načítáním počtu pulsů a stopnutím činnosti po čase, který potřebuje pro přejetí na novou polohu. K tomu bylo vytvořeno dynamické počítadlo, které v závislosti na rozdílu mezi novým a předchozím úhlem otočení spočítá nutný čas k jeho dosažení. Funguje na principu počítání vzorků, neboť jsou známá vzorkovací frekvence a počet stupňů otočení za daný čas. Jednoduše řečeno, čím větší úhel otočení, tím adekvátně delší čas je servu poskytnut.

Jakmile je zaznamenána změna na vstupu, přepínač vyresetuje počítadlo a znovu pustí signál, pokud ke změně nedojde, počítadlo po stanoveném počtu vzorků samo přeruší signál do serva. Velikost 1 vzorku byla stanovena dle PWM hodnot na 1.10^{-4} s. Výrobce v dokumentaci udává otočení serva o 60° za čas 0,12 s, takže pro 180° zaokrouhleno na 0,5 s kvůli zohlednění výrobní tolerance, navíc jde o hodnotu nezatíženého serva. Požadavek je, aby úhlu otočení bylo určitě dosaženo před přerušením signálu. Spočítá se, o kolik se změnila délka pulsu, když za plný čas 0,5 s proběhne 5000 vzorků (100 %). Tím že celý proces je převáděn na procenta, je důležitým údajem počet vzorků *(samples)* za 1 % času celkového pohybu.
Výsledkem je počet vzorků jako rozdíl mezi novou a původní hodnotou otočení ve stupních.

6.5 Řešení 3D polohování kamery v Simulinku

Výsledné řešení v příloze 15.

Počáteční část procesu, kdy obraz (RGB) z RPi kamery prochází detekcí objektu až po analýzu zjištěného *BW* obrazu (*Blob Analysis)*, je identická s příkladem 2. Na to ale navazuje výpočet sférických souřadnic s nalezením kritické oblasti, v závislosti

na pohybu objektu je zde připojení přes PWM na výstup do GPIO na ovládání 2 serv a nakonec výpis souřadnic na monitor.

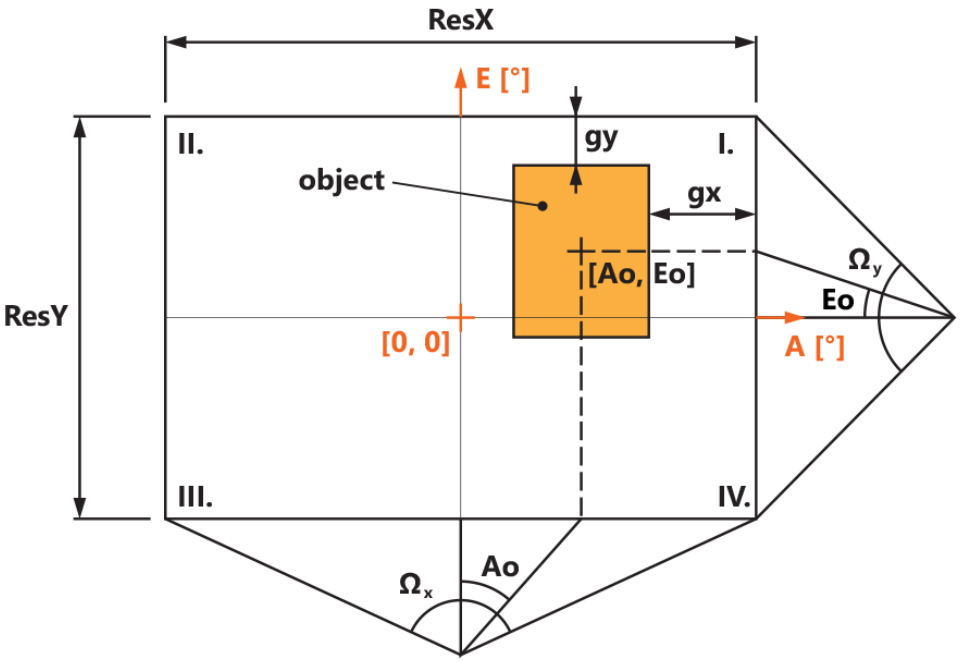
Výpočet *A* a *E* pro pohyb serva

(Řešení v příloze 13.)

Procedura je v bloku *AECalc (Azimuth & Elevation)* a počítá úhly otočení serva v závislosti na pohybu detekovaného objektu. Postup se skládá z těchto kroků:

- Výpočet souřadnic středu obrazu dle rozlišení a středu objektu
- Zjištění, ve kterém kvadrantu leží střed objektu
- Hledání největšího přiblížení a zjištění porušení kritické vzdálenosti
- Výsledná změna úhlů pro otočení serv

Funkce tohoto bloku je otočení kamery tak, aby objekt byl stále v záběru, pokud se přiblíží na kritickou vzdálenost okraji obrazovky (obr. 30). Souřadnice středu kamery výchází z konstant snímaného rozlišení *(ResX, ResY)*, střed objektu je znám, protože souřadnice detekované oblasti přijdou z Blob analýzy ve formě 1 x 4 matice *(CoordsArea)*.



Obr. 30: Znázornění proměnných pro výpočet sférických souřadnic objektu.

Dále se zjistí, ke kterému okraji obrazovky je objekt nejvíce přiblížen. To znamená určení kvadrantu *(I. - IV.)*, do kterého spadá střed objektu. Pak už se vzdálenosti příslušných stran objektu ke kraji obrazovky *(gx, gy)* dopočítají a porovná se, zda tato hodnota porušuje max. nastavený limit *(CritDist)*, což je hodnota přiblížení objektu ke kraji, kdy začne reagovat servo a kompenzovat tuto výchylku.

Výsledkem jsou souřadnice úhlů pro horizontální *(A)* a vertikální *(E)* otočení serv. Pro přechod signálu do ovládání serva je ještě potřeba zohlednit rozdílné vzorkovací

frekvence, kdy kamera snímá např. 1/20 s, zatímco PWM serva pracuje řádově jinde na hodnotě 1.10^{-4} s. Tento rozdíl kompenzuje blok *Rate Transition*.

Globální sférický souřadnicový systém

(Řešení v příloze 12.)

Výsledné reálné hodnoty azimutu a elevace jsou kombinací hodnot otočení serva a lok. pozice objektu na monitoru. Toto zajistí blok *ViewCoordsAE (A and E angle calc)*:

- Výpočet souřadnic středu obrazu dle rozlišení a středu objektu
- Převod úhlů otočení *A* a *E* serva 0° - 180° na 0° - 360°
- Výpočet lokálních souřadnic objektu [*Ao, Eo*] (obr. 30)
- Kombinace *A, E* serva a *Ao, Eo* objektu
- Výsledek převeden na globální sférický souřadnicový systém

Souřadnice středu obrazu a středu objektu se získají stejně jako v předchozím případě. Serva pracují v rozsahu 0° - 180° a je nutné jejich hodnoty převést na systém 0° - 360°. Pro určení sférických souřadnic objektu se využije znalosti zorného úhlu kamery v obou směrech.

Výsledné hodnoty se zobrazují v celých stupních a po přejetí z pravé strany doleva přes počátek [0, 0] musí azimut vypsat číslo 359 a dále klesat, tedy dodržet správný systém souřadnic, jak je vyznačeno zeleně na obr. 27. Např. při otočení kamery zcela vlevo bude hodnota azimutu ve středu obrazu 270°. Elevace funguje analogicky.

7 ZÁVĚR

V prvním demonstračním příkladu, který popisuje praktické možnosti využití Raspberry Pi, probíhal výpočet lokalizace zdroje zvuku s připojením externí zvukové karty a stereo mikrofonem. Pro výpočet byla použita metoda výpočtu časového zpoždění. Nejdříve byl otestován model s aplikovanou funkcí křížové korelace a při druhé verzi použit předdefinovaný blok z knihovny Simulinku přímo pro výpočet úhlu.
Porovnáním obou výsledků bylo sice zjištěno, že v obou případech jsou hodnoty podobně neurčité, ale vzhledem k široké problematice akustiky jde o více ovlivňujících faktorů velmi závislých na kvalitě mikrofonů, prostředí nebo šumu v signálu. Pro ověření hodnot byl použit ještě druhý mikrofon se stejným výsledkem.

Ve druhém příkladu modelové schéma sledování objektu při testování úspěšně fungovalo podle očekávání a při změně typu barvy či posunutí prahu citlivosti správně reagovala detekce. Použití této metody je ale závislé na okolním osvětlení, intenzitě a barevném spektru detekovaného objektu.

Kalibrační proces pracoval velice přesně nejen při měření rozměrů objektu, ale i při výpočtu z-souřadnice. Velkou výhodou je schopnost tolerování rozměrových nedokonalostí kalibračního tělesa, navíc výsledek je ihned vidět na monitoru a v případě potřeby lze kalibraci kdykoliv zopakovat.

Pro funkčnost polohování kamery u třetího příkladu se podařilo vyřešit zásadní překážku pro ovládání serva. Chybějící blok Simulinku byl nahrazen vlastním programovým schématem, mikroserva experimentálně otestována pro nalezení PWM charakteristiky. Mechanika polohování je plně funkční.

Původní souřadnicový systém byl nahrazen sférickým a pracuje správně pro pohyb objektu po obrazovce s kombinací otáčení kamery v prostoru. Nedostatkem může být chybějící souřadnice vzdálenosti, která ale s použitým vybavením nelze zjistit, nebo malá snímkovací frekvence Raspberry Pi kamerky, což se dá vyřešit výměnou za výkonnější zařízení.

8 SEZNAM POUŽITÉ LITERATURY

[1] UPTON Eben; HALFACREE Gareth: Raspberry Pi, COMPUTER PRESS, 2016. Martin Stříž, Bučovice, 2015. 280 s. ISBN: 978-80-251-4819-8

[2] Raspberry Pi® User Guide. *University of North Carolina Asheville* [online]. UK, 2012 [cit. 2017-05-14]. Dostupné z: http://www.cs.unca.edu/~bruce/Fall14/360/RPiUsersGuide.pdf

[3] FAQS. *RASPBERRY PI FOUNDATION* [online]. [2012] [cit. 2017-05-15]. Dostupné z: https://www.raspberrypi.org/help/faqs/

[4] Co je nového u Raspberry Pi? Všechno! *ROOT.CZ* [online]. Praha: Internet Info, 2015 [cit. 2017-05-15]. Dostupné z: https://www.root.cz/clanky/co-je-noveho-u-raspberry-pi-vsechno/

[5] RASPBERRY PI MODEL A 256MB RAM. *Adafruit* [online]. NYC: MIT hacker & engineer, [2012] [cit. 2017-05-15]. Dostupné z: https://www.adafruit.com/product/1344

[6] RASPBERRY PI MODEL A & B: BOARDS. *Adafruit* [online]. NYC: MIT hacker & engineer, 2012 [cit. 2017-05-15]. Dostupné z: https://www.adafruit.com/category/379

[7] Raspberry Pi Model A+ 512MB RAM. *Adafruit* [online]. NYC: MIT hacker & engineer, 2016 [cit. 2017-05-15]. Dostupné z: https://www.adafruit.com/product/2266

[8] PHILLIP71NEWMAN (nick). Raspberry Pi Model A drawings and dimensions. *RASPBERRY PI FOUNDATION* [online]. 2013 [cit. 2017-04-16]. Dostupné z: https://www.adafruit.com/product/2266

[9] RASPBERRY PI MODEL A+: CHEAPER, SMALLER, CURVIER; THE ALL NEW RASPBERRY PI. *Linux Voice* [online]. 2014 [cit. 2017-04-18]. Dostupné z: https://www.linuxvoice.com/raspberry-pi-model-a/

[10] Raspberry Pi. *Wikipedia, The Free Encyclopedia* [online]. Wikimedia Foundation [cit. 2017-04-12]. Dostupné z: https://en.wikipedia.org/wiki/Raspberry_Pi

[11] abishur (nick). Getting Started with the Raspberry Pi. *RASPBERRY PI FOUNDATION* [online]. 2012 [cit. 2017-04-16]. Dostupné z: https://www.raspberrypi.org/forums/viewtopic.php?t=4751

[12] RASPBERRY PI MODEL B+. *Adafruit* [online]. NYC: MIT hacker & engineer [cit. 2017-05-15]. Dostupné z: https://www.adafruit.com/product/1914

[13] RASPBERRY PI 1 MODEL B+. *RASPBERRY PI FOUNDATION* [online]. 2014 [cit. 2017-04-17]. Dostupné z: https://www.raspberrypi.org/products/model-b-plus/

[14] RASPBERRY PI RASPBERRY PI ZERO. *RASPBERRY PI FOUNDATION* [online]. 2015 [cit. 2017-04-17]. Dostupné z: https://www.raspberrypi.org/documentation/hardware/raspberrypi/mechanical/rpi-zero-v1_2_dimensions.pdf

[15] A Tour of the Pi Zero. *Adafruit* [online]. NYC: MIT hacker & engineer, 2015 [cit. 2017-04-20]. Dostupné z: https://learn.adafruit.com/introducing-the-raspberry-pi-zero?view=all

[16] RASPBERRY PI ZERO: THE $5 COMPUTER. *RASPBERRY PI FOUNDATION* [online]. 2015 [cit. 2017-04-20]. Dostupné z: https://www.raspberrypi.org/blog/raspberry-pi-zero/

[17] Raspberry Pi Zero W. *The PiHut* [online]. Mann Enterprises [cit. 2017-04-20]. Dostupné z: https://thepihut.com/products/raspberry-pi-zero-w

[18] RASPBERRY PI 3 MODEL B. *RASPBERRY PI FOUNDATION* [online]. 2016 [cit. 2017-05-15]. Dostupné z: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

[19] OLŠAN, Jan. Raspberry Pi 3 trpí přehříváním. Teploty CPU jdou nad 80°C a výš i bez skříně. *Cnews.cz* [online]. Praha: Mladá fronta, 2016 [cit. 2017-05-15]. Dostupné z: https://www.cnews.cz/raspberry-pi-3-trpi-prehrivanim-teploty-cpu-jdou-nad-80c-a-vys-i-bez-skrine/

[20] Raspberry Pi mění svět: Seznamte se s nejzajímavějším počítačem dneška. *Aktuálně.cz: DVTV* [online]. Praha: Economia, 2016 [cit. 2017-05-15]. Dostupné z: https://video.aktualne.cz/tech-news/raspberry-pi-meni-svet-seznamte-se-s-nejzajimavejsim-pocitac/r~8df32710e3ea11e584160025900fea04/?redirected=1494857333

[21] VÍTEK, Jan. Raspberry Pi 3 je tu, co nabídne? *Svět hardware* [online]. oXyMedia, 2016. [cit. 2017-04-21]. Dostupné z: http://www.svethardware.cz/raspberry-pi-3-je-tu-co-nabidne/41920

[22] Raspberry Pi to Arduino Shields Connection Bridge. *Cooking hacks* [online]. [cit. 2017-04-21]. Dostupné z: https://www.cooking-hacks.com/documentation/tutorials/raspberry-pi-to-arduino-shields-connection-bridge#step6

[23] HORÁČEK, Petr. Linuxsoft.cz. *Raspberry π* [online]. Praha: Pavel Kysilka, 2012 [cit. 2017-04-26]. ISSN 1801-3805. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=1953

[24] GPIO: RASPBERRY PI MODELS A AND B. *RASPBERRY PI FOUNDATION* [online]. [cit. 2017-05-16]. Dostupné z: https://www.raspberrypi.org/documentation/usage/gpio/

[25] Raspberry Pi 3 Pinout Model B, RPI2, B+ 40Pin GPIO Pinout. *Myelectronicslab.com* [online]. 2016 [cit. 2017-05-15]. Dostupné z: https://www.myelectronicslab.com/tutorial/raspberry-pi-3-gpio-model-b-block-pinout/

[26] Banana Pi R1: malý počítač se SATA, Wi-Fi a switchem. *Root.cz* [online]. Praha: Petr Krčmář, 2015 [cit. 2017-05-15]. Dostupné z: https://www.root.cz/clanky/banana-pi-r1-maly-pocitac-se-sata-wi-fi-a-switchem/

[27] BeagleBone black, víc než jen alternativa k Raspberry PI. *Robodoupe.cz* [online]. Praha: HOBBYROBOT, 2014 [cit. 2017-05-15]. Dostupné z: http://robodoupe.cz/2014/beaglebone-black-vic-nez-jen-alternativa-k-raspberry-pi/

[28] BeagleBone Black: What is BeagleBone Black? *Beagleboard.org* [online]. 2017 [cit. 2017-04-18]. Dostupné z: https://beagleboard.org/black

[29] DOLEŽAL, Jakub. Raspberry Pi, Orange Pi, Banana Pi: který jako HTPC? *Tvfreak.cz* [online]. oXy Online, 2015 [cit. 2017-05-15]. ISSN 1802-1328. Dostupné z: https://www.tvfreak.cz/raspberry-pi-orange-pi-banana-pi-ktery-jako-htpc/5533-4

[30] SOPINE 64-bit Compute Mobile: Simple, Expendable & powerful. *Pine64.org* [online]. 2017 [cit. 2017-05-15]. Dostupné z: https://www.pine64.org/?page_id=1491

[31] CO JE TO ARDUINO? *ARDUINO.CZ* [online]. 2017 [cit. 2017-05-15]. Dostupné z: https://arduino.cz/co-je-to-arduino/

[32] NOOBS. *Raspberrypi.org* [online]. RASPBERRY PI FOUNDATION, 2017 [cit. 2017-05-15]. Dostupné z: https://www.raspberrypi.org/downloads/noobs/

[33] PRAVDA, Michal. Instalace Raspberry PI 3 – NOOBS. *Raspishop.cz* [online]. 2016 [cit. 2017-05-15]. Dostupné z: http://www.raspishop.cz/2016/03/instalace-raspberry-pi-3-noobs/

[34] INTRODUCING RASPBERRY PI HATS. *Raspberrypi.org* [online]. RASPBERRY PI FOUNDATION, 2014 [cit. 2017-05-15]. Dostupné z: https://www.raspberrypi.org/blog/introducing-raspberry-pi-hats/

[35] Raspberry Pi - Příslušenství: Rozšiřující moduly. *RPiShop.cz* [online]. České Budějovice: Michal Prenner [cit. 2017-05-15]. Dostupné z: http://rpishop.cz/12-rozsirujici-moduly

[36] KOTEK, Lukáš. Raspberry Pi: Raspberry Pi a možnosti jeho praktického použití. *RVP.CZ* [online]. 2012 [cit. 2017-05-15]. ISSN 1802-4785. Dostupné z: http://spomocnik.rvp.cz/clanek/16825/RASPBERRY-PI-A%C2%A0MOZNOSTI-JEHO-PRAKTICKEHO-POUZITI.html

[37] *MATLAB/SIMULINK* [online]. The MathWorks, USA, c1994-2017 [cit. 2017-05-15]. Dostupné z: https://www.mathworks.com/

[38] *MATLAB & SIMULINK* [online]. Humusoft, USA, c1991-2017 [cit. 2017-05-15]. Dostupné z: http://www.humusoft.cz/matlab/

[39] Modelování elektromechanické soustavy v prostředí Matlab a Simulink (část 1). *Automa.cz* [online]. Automa, c2016, **2008**(08) [cit. 2017-05-15]. Dostupné z: http://automa.cz/cz/casopis-clanky/modelovani-elektromechanicke-soustavy-v-prostredi-matlab-a-simulink-cast-1-2008_08_37720_6045/

[40] Raspberry Pi Programming with MATLAB and Simulink: Build Raspberry Pi projects using high-level programming and block diagrams. *MathWorks* [online]. USA, c1994-2017 [cit. 2017-05-15]. Dostupné z: https://www.mathworks.com/discovery/raspberry-pi-programming-matlab-simulink.html

[41] Hardware Support: Raspberry Pi Support from Simulink. *MathWorks* [online]. USA, c1994-2017 [cit. 2017-05-15]. Dostupné z: https://www.mathworks.com/hardware-support/raspberry-pi-simulink.html

[42] ČEVELA, Lubomír. Jak moc hřeje Raspberry Pi 3? *LinuxEXPRES* [online]. CCB, 2016 [cit. 2017-05-15]. Dostupné z: https://www.linuxexpres.cz/hardware/jak-moc-hreje-raspberry-pi-3

[43] SD Card Formatter. *SD Association* [online]. SD Association, c2017 [cit. 2017-05-15]. Dostupné z: https://www.sdcard.org/downloads/formatter_4/

[44] Videos and Webinars: Install the MATLAB Support Package for Raspberry Pi. *MathWorks* [online]. USA, c1994-2017 [cit. 2017-05-15]. Dostupné z: https://www.mathworks.com/videos/install-the-matlab-support-package-for-raspberry-pi-94266.html?s_tid=srchtitle

[45] ŠIROKÝ, Michal. *Základy práce s programem Simulink* [online]. 2007 [cit. 2017-05-15]. Dostupné z: http://www.michalsiroky.com/control/materialy/simulink/navod_na_simulink.pdf

[46] PEASLEY, Eric. *An Introduction to Using Simulink* [online]. University of Oxford: Department of Engineering Science, 2013 [cit. 2017-05-15]. Dostupné z: http://www.eng.ox.ac.uk/~labejp/Seminar/Simulink/Simulink_Introduction.pdf

[47] KRISHNAMURTHY, Ashok a Siddharth SAMSI. *SC 09 Education Program: Simulink Basics for Engineering Applications* [online]. Ohio, USA, 2009 [cit. 2017-05-15]. Dostupné z: http://teachers.teicm.gr/anastasiou/wp-content/uploads/2014/11/091114-Simulink-Basics.pdf

[48] Simulink Support Package for Raspberry Pi Hardware: Getting Started with Simulink Support Package for Raspberry Pi Hardware. *MathWorks* [online]. USA, c1994-2017 [cit. 2017-05-15]. Dostupné z: https://www.mathworks.com/help/supportpkg/raspberrypi/examples/getting-started-with-raspberry-pi-hardware.html

[49] MATLAB Support Package for Raspberry Pi Hardware: Getting Started with MATLAB Support Package for Raspberry Pi Hardware. *MathWorks* [online]. USA, c1994-2017 [cit. 2017-05-15]. Dostupné z: https://www.mathworks.com/help/supportpkg/raspberrypiio/examples/getting-started-with-matlab-support-package-for-raspberry-pi-hardware.html#zmw57dd0e120

[50] TKADLEC, Josef. *LOKALIZACE VZDÁLENÉHO ZDROJE ZVUKU POLEM MIKROFONŮ* [online]. 2011 [cit. 2017-05-15]. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=40757. Bakalářská práce. VUT FEKT. Vedoucí práce Ing. Zdeněk Havránek, Ph.D.

[51] Interaural time difference. *Wikipedia, The Free Encyclopedia* [online]. Wikimedia Foundation [cit. 2017-05-15]. Dostupné z: https://en.wikipedia.org/wiki/Interaural_time_difference

[52] Understanding Ocean Acoustics. *Ocean Explorer* [online]. NOAA Ocean Explorer Webmaster, 2015 [cit. 2017-05-15]. Dostupné z: http://oceanexplorer.noaa.gov/explorations/sound01/background/acoustics/acoustics.html

[53] Xcorr: Cross-correlation. *MathWorks* [online]. USA, c1994-2017 [cit. 2017-05-15]. Dostupné z: https://www.mathworks.com/help/signal/ref/xcorr.html#bub68m7

[54] MECHANICKÉ KMITÁNÍ A VLNĚNÍ: Rychlost zvuku. *Encyklopedie fyziky* [online]. Jaroslav Reichl, Martin Všetička, c2006-2017 [cit. 2017-05-15]. Dostupné z: http://fyzika.jreichl.com/main.article/view/189-rychlost-zvuku

[55] MURRAY, John C., Harry ERWIN a Stefan WERMTER. *Robotic Sound-Source Localization and Tracking Using Interaural Time Difference and CrossCorrelation* [online]. Sunderland, SR6 0DD [cit. 2017-05-15]. Dostupné z: https://www.ent.mrt.ac.lk/~rohan/teaching/EN5101/Reading/murray_ulm04.pdf. University of Sunderland.

[56] Phased Array System Toolbox. *MathWorks* [online]. USA, c1994-2017 [cit. 2017-05-15]. Dostupné z: https://www.mathworks.com/help/phased/index.html

[57] Barevný model. *Wikipedia, The Free Encyclopedia* [online]. Wikimedia Foundation, 2016 [cit. 2017-05-15]. Dostupné z: https://cs.wikipedia.org/wiki/Barevn%C3%BD_model

[58] Image Processing Toolbox: Perform image processing, analysis, and algorithm development. *MathWorks* [online]. USA, c1994-2017 [cit. 2017-05-15]. Dostupné z: https://www.mathworks.com/products/image.html

[59] MATLAB Support Package for Raspberry Pi Hardware: Servo. *MathWorks* [online]. USA, c1994-2017 [cit. 2017-05-15]. Dostupné z: https://www.mathworks.com/help/supportpkg/raspberrypiio/ref/servo.html

[60] Rpi Camera Module: Rpi Camera Module - Pi NoIR v1. *Embedded Linux Wiki* [online]. Bill Traynor, 2016 [cit. 2017-05-15]. Dostupné z: http://elinux.org/Rpi_Camera_Module

[61] Configure Models with Pulse Width Modulation (PWM) Signals. *MathWorks* [online]. USA, c1994-2017 [cit. 2017-05-15]. Dostupné z: https://www.mathworks.com/help/slcontrol/ug/models-with-pulse-width-modulation-pwm-signals.html

[62] ING. ČERNÝ, Michal. *Seriál o servech* [online]. [cit. 2017-05-15]. Dostupné z: http://www.modelklubnachod.wz.cz/serial%20o%20servech%20ing,cerny.pdf

9 SEZNAM ZKRATEK

ALSA	Advanced Linux Sound Architecture
BW	Black and White
CCTV	Closed Circuit Television
CSI	Camera Serial Interface
DOA	Direction of Arrival
GCC-PHAT	Generalized CrossCorrelator with Phase Transform
GPIO	General Purpose Input/Output
GPU	Graphic Processing Unit
HAT	Hardware Attached on Top
HDD	Hard disk drive
HDMI	High-Definition Multimedia Interface
hw	hardware
I2C	Inter-Integrated Circuit
I2S	Integrated Interchip Sound
IDE	Integrated Development Environment
IoT	Internet of Things
LAN	Local Area Network
LED	Light Emitting Diode
NOOBS	New Out Of the Box Software
PWM	Pulse Width Modulation
px	pixel
RAM	Random Access Memory
RC	Radio Controlled
RGB	Red, Green, Blue
RPi	Raspberry Pi
RTC	Real Time Clock
SoC	System on a Chip
SPI	Serial Peripheral Interface
TDOA	Time Delay Of Arrival
TOA	Time Of Arrival
UDP	User Datagram Protocol
ULA	Uniform Linear Array

10 SEZNAM PŘÍLOH

PŘÍKLAD 1: URČENÍ SMĚRU ZDROJE ZVUKU

Příloha 1: Hlavní schéma.

Příloha 2: Zdrojový kód *LagCalc (Stereo Splitting and lag calculation):*
Zdrojový kód *DirCalc (Direction Calculation):*

PŘÍKLAD 2: SLEDOVÁNÍ OBJEKTU

Příloha 3: Hlavní schéma.

Příloha 4: Subsystém *Object Tracking*.

Příloha 5: Subsystém *Highlight Area and Text on the Screen*.

Příloha 6: Zdrojový kód *TCalc (Tracking Calculation).*
Zdrojový kód *ViewCoords (CenterPoint).*

Příloha 7: Subsystém *Calibration*.

Příloha 8: Zdrojový kód *RealCoords (Ratio for real coords calculation).*
Zdrojový kód *ZCalc (Z distance calculation).*

PŘÍKLAD 3: 3D POLOHOVÁNÍ KAMERY

Příloha 9: Hlavní schéma.

Příloha 10: Subsystém *A nad E on the Screen*.

Příloha 11: Subsystém *Servo horizontal (Servo vertical)*.

Příloha 12: Zdrojový kód *ViewCoordsAE (A and E angle calc).*

Příloha 13: Zdrojový kód *AECalc (Azimuth & Elevation Calculation).*

Příloha 14: Subsystém *Interpolation to degrees*.
Subsystém *Convert degrees to percent*.

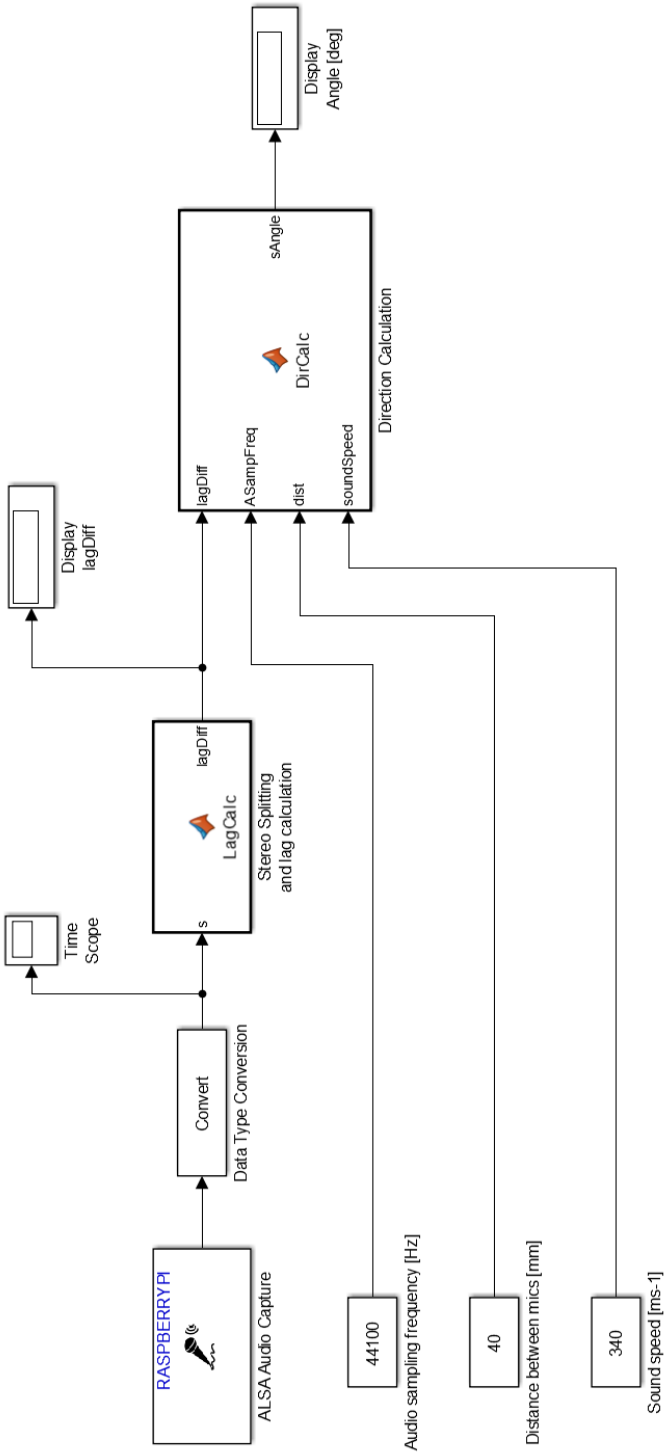
Příloha 15: Zdrojový kód *PStop (Stop after positioning).*

Elektronická příloha: Simulink soubory řešených příkladů

Příloha 1:

Př. 1) Určení směru zdroje zvuku

Hlavní schéma.



Příloha 2:
Př. 1) Určení směru zdroje zvuku

Zdrojový kód *LagCalc (Stereo Splitting and lag calculation):*

```
function lagDiff = LagCalc(s)
%{
Calculation the lag using cross correlation (xcorr)
s ..... stereo signal
lagDiff ..... count of samples diff. between signals
-----------------------------------------
%}
% Left Channel
Lch = s(:,1);
% Right Channel
Rch = s(:,2);

% cross correlation to find peak->lag
[r,lag] = xcorr(Lch, Rch);
[~,i] = max(abs(r));
lagLR = lag(i);

[r,lag] = xcorr(Rch, Lch);
[~,i] = max(abs(r));
lagRL = lag(i);

% find which signal has lag
if lagLR > lagRL
    lagDiff = lagLR;
else
    lagDiff = lagRL;
end;
```

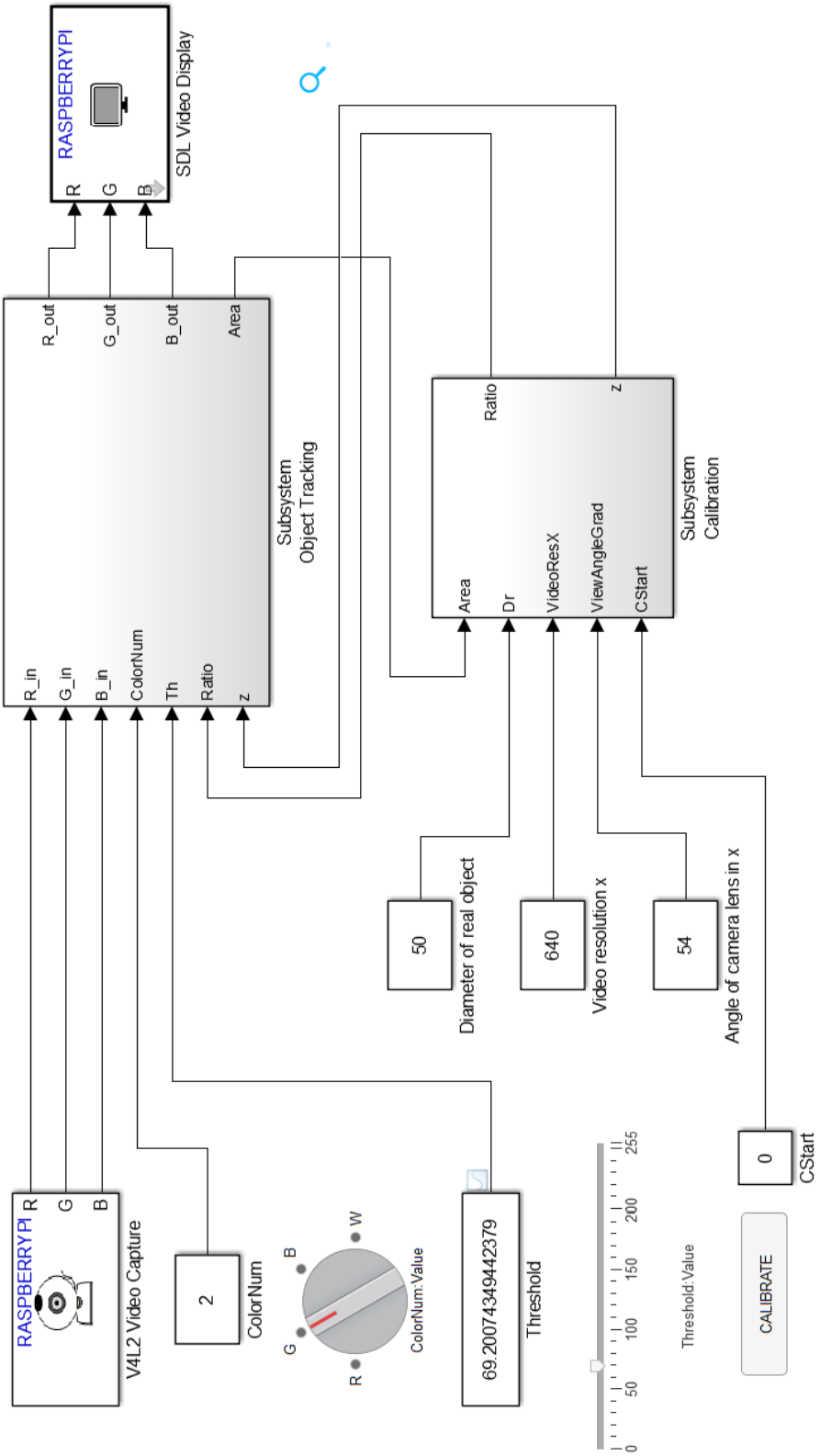
Zdrojový kód *DirCalc (Direction Calculation):*

```
function sAngle = DirCalc(lagDiff, ASampFreq, dist, soundSpeed)
%{
Calculation the direction of arrival signal (DOA)
lagDiff ..... count of samples diff. between signals
ASampFreq ..... audio sampling frequency
dist ..... distance L and R mic
soundSpeed ..... sound speed
sAngle ..... result angle of arrival signal
-----------------------------------------
%}
% distance mm -> m
dist = dist / 1000;

if lagDiff ~= 0
    x = lagDiff / ASampFreq * soundSpeed;
    alfa = asin(x / dist) * 180 / pi;
else
    alfa = 0;
end

sAngle = alfa;
```

Příloha 3:

Př. 2) Sledování objektu

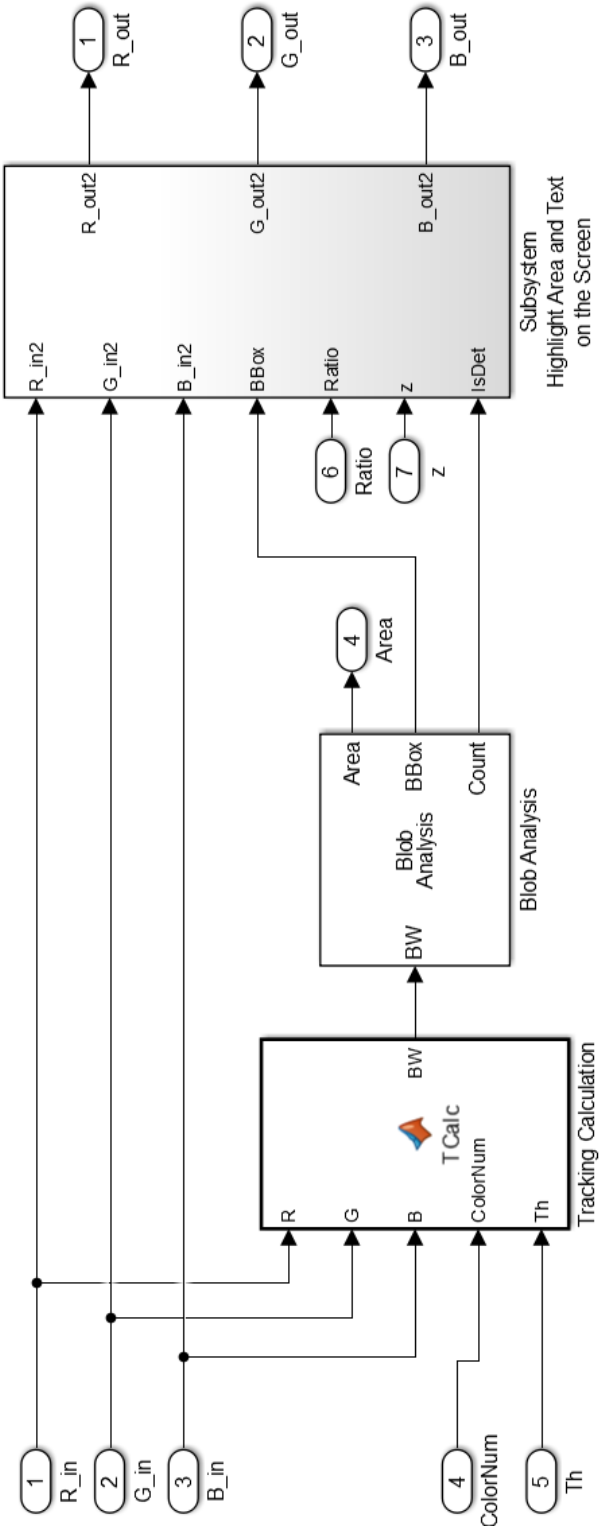
Hlavní schéma.



Příloha 4:

Př. 2) Sledování objektu

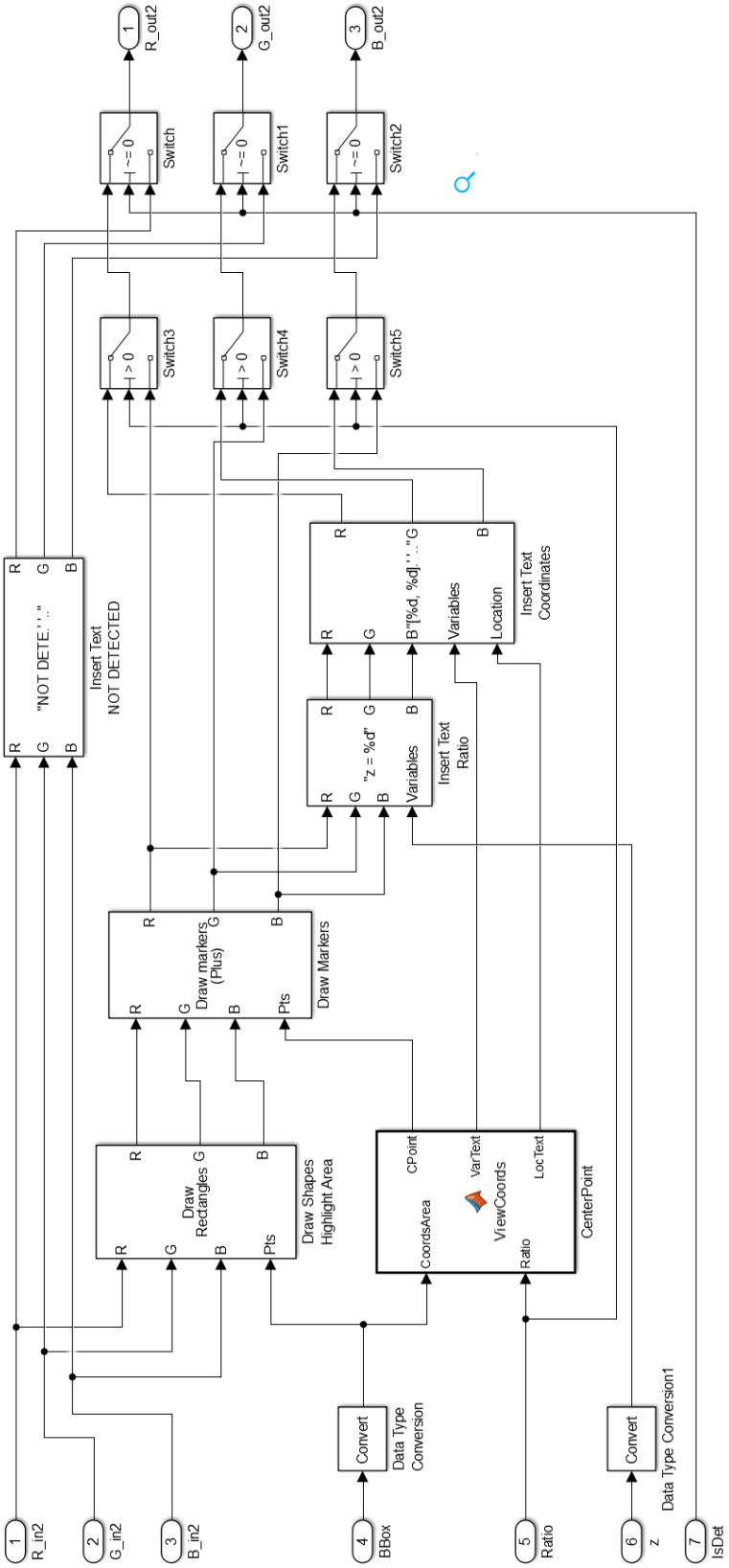
Subsystém *Object Tracking*.



Příloha 5:

Př. 2) Sledování objektu

Subsystém *Highlight Area and Text on the Screen*.



Příloha 6:
Př. 2) Sledování objektu

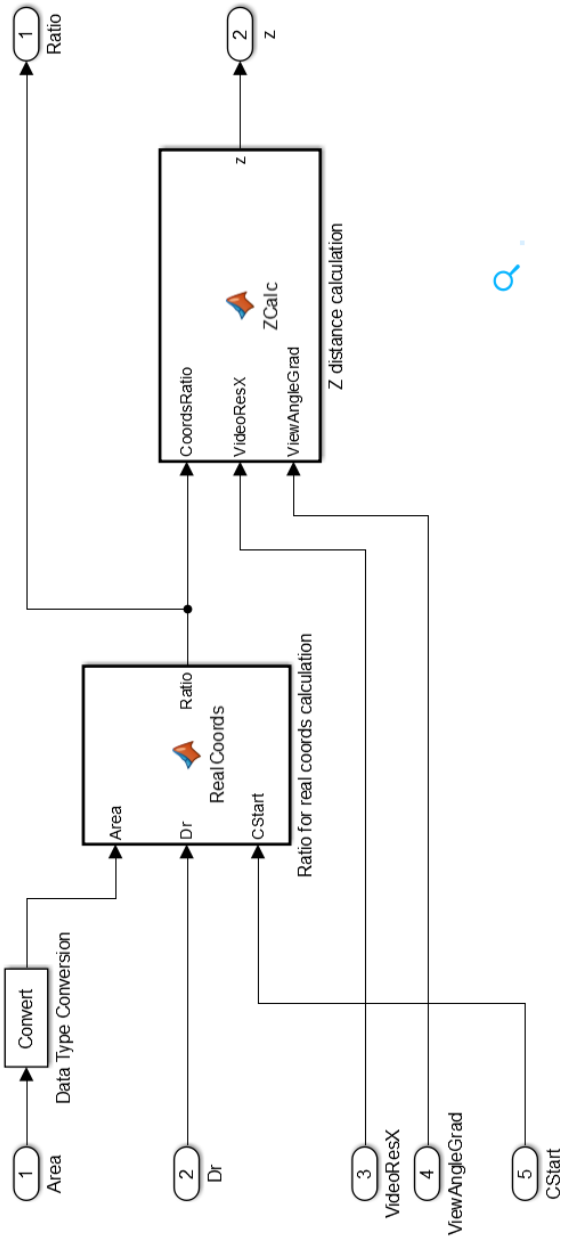
Zdrojový kód *TCalc (Tracking Calculation).*

```
function BW = TCalc(R, G, B, ColorNum, Th)
% C - TrackingColor
% BW is matrix of nums 1 if TrackingColor > thresh, else 0
switch ColorNum
    case 1                      % ColorNum is R
        C = R - G/3 - B/4;
        C = C > Th;
    case 2                      % ColorNum is G
        C = G - R/4 - B/4;
        C = C > Th;
    case 3                      % ColorNum is B
        C = B - R/4 - G/3;
        C = C > Th;
    case 4                      % ColorNum is White
        C = R > Th & G > Th & B > Th;
        C = imfill(C,'holes');
    otherwise
        C = uint8(zeros(size(R)));
        C = C > Th;
end
BW = C;
```

Zdrojový kód *ViewCoords (CenterPoint).*

```
function [CPoint, VarText, LocText] = ViewCoords(CoordsArea, Ratio)
% CoordsArea - vector [1x4]
x1 = CoordsArea(2);
y1 = CoordsArea(1);
x2 = CoordsArea(4);
y2 = CoordsArea(3);

% center of area
xc = x1 + x2 / 2;
yc = y1 + y2 / 2;
CPoint = [yc xc];

% variables - coordinates of det.area on screen - in CPoint
% xc, yc - center point; x2, y2 - width, height
VarText = round([xc yc x2 y2] * Ratio);

% location of VarText
LocText = [x1 y1+y2+2];
```

Příloha 7:
Př. 2) Sledování objektu
Subsystém *Calibration*.



Příloha 8:
Př. 2) Sledování objektu

Zdrojový kód *RealCoords (Ratio for real coords calculation).*

```
function Ratio = RealCoords(Area, Dr, CStart)
% Dc .. diameter of calibration object in pc
% Dr .. real diameter of the calibration object
% CStart .. calibration start

persistent r;
if isempty(r)
    r = 0;
end
if (CStart == 1)
    Dc = sqrt(4 * Area / pi);
    r = Dr / Dc;
end
Ratio = r;
```

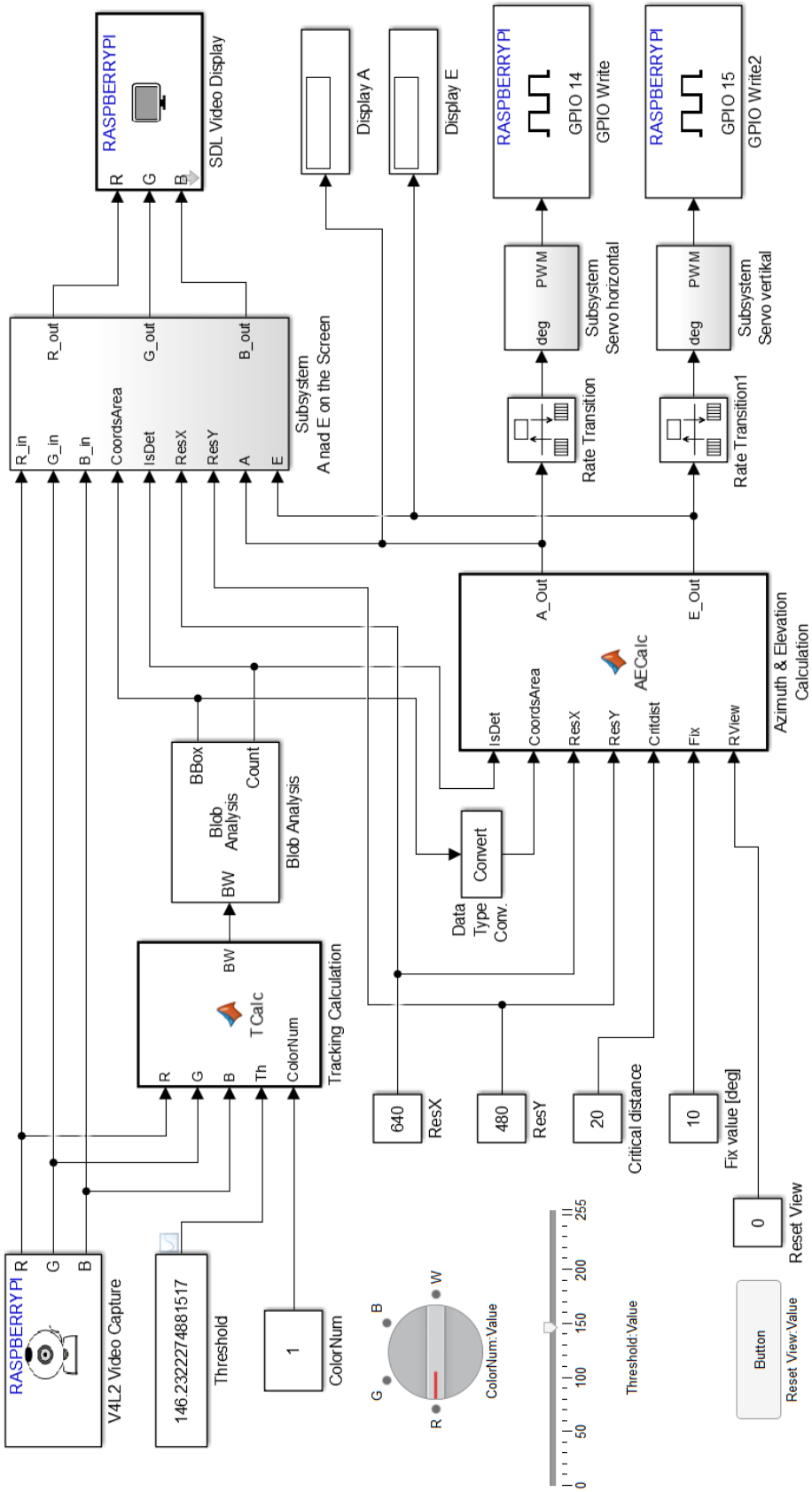
Zdrojový kód *ZCalc (Z distance calculation).*

```
function z = ZCalc(CoordsRatio, VideoResX, ViewAngleGrad)
% z .. calculating the distance between camera and object
% ViewAngle .. angle of camera lens
% VideoResX .. max width of video picture
ViewAngleRad = ViewAngleGrad * pi / 180;
zPix = VideoResX / (2 * tan(ViewAngleRad / 2));
z = zPix * CoordsRatio;
```

Příloha 9:

Př. 3) 3D polohování kamery

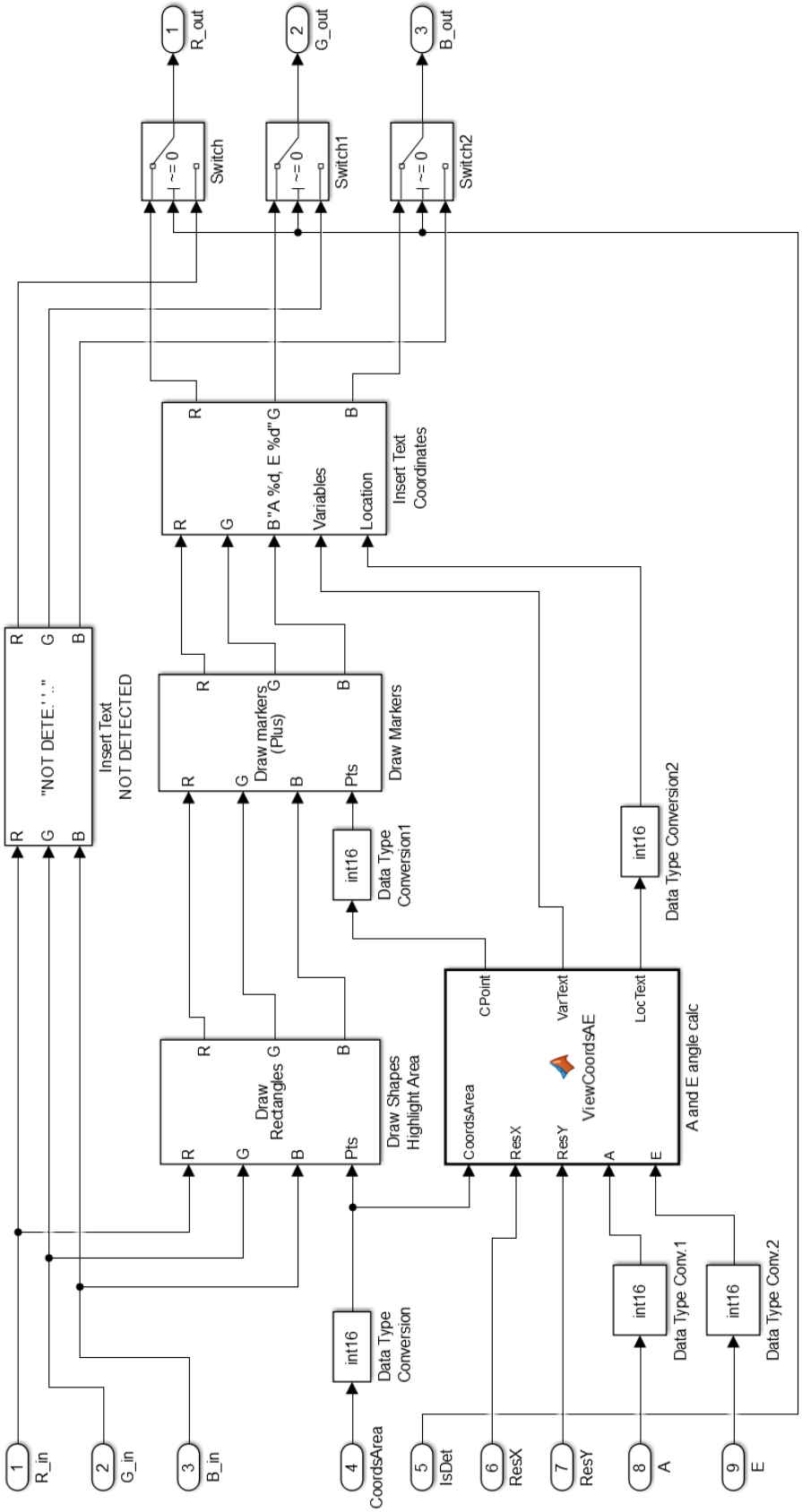
Hlavní schéma.



Příloha 10:

Př. 3) 3D polohování kamery

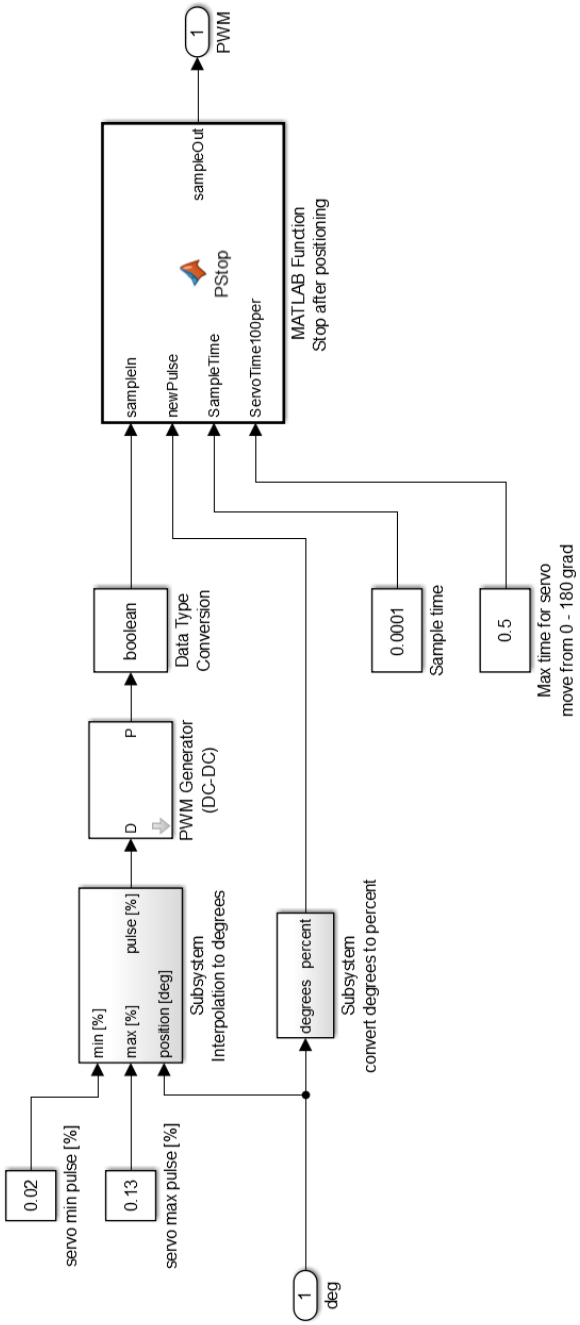
Subsystém *A and E on the Screen*.



Příloha 11:

Př. 3) 3D polohování kamery

Subsystém *Servo horizontal (Servo vertical)*.



Příloha 12:
Př. 3) 3D polohování kamery

Zdrojový kód *ViewCoordsAE (A and E angle calc).*

```
function [CPoint, VarText, LocText] = ViewCoordsAE(CoordsArea, ResX,
ResY, A, E)
%{
CoordsArea ..... coordinates of area - get from BBox
ResX, ResY ..... camera resolution x and y
A E ..... azimut and elevation angle of camera
CPoint ..... center point of object
VarText ..... variables to display on screen
LocText ..... location of variables on screen
Ao Eo ..... azimut and elevation angle of the object
ViewAngleX, ViewAngleX ..... angle view of camera in x, y
scrCX, scrCY ..... center point of screen
----------------------------------------
%}
% center point of screen
scrCX = ResX / 2;
scrCY = ResY / 2;
% angle view of camera in x, y
ViewAngleX = 54;
ViewAngleY = 41;
% CoordsArea - vector [1x4]
x1 = CoordsArea(2);
y1 = CoordsArea(1);
w = CoordsArea(4);
h = CoordsArea(3);
% center of area
xc = x1 + w / 2;
yc = y1 + h / 2;
CPoint = [yc xc];
% convert A and E from servo deg to spherical system
if A >= 90
    A = 90 - A;
else
    A = 360 - (A - 90);
end
E = E - 90;
% local Ao and Eo of object on screen
Ao = ViewAngleX / ResX * (xc - scrCX);
Eo = ViewAngleY / ResY * (scrCY - yc);
% global Ao and Eo = local on screen + servo position
Ao = A + Ao;
Eo = E + Eo;
if Ao < 0
    Ao = 360 - abs(Ao);
end
if Ao >= 360
    Ao = Ao - 360;
end

VarText = [Ao Eo];
% location of VarText
LocText = [x1 y1+h+2];
```

Příloha 13:
Př. 3) 3D polohování kamery

Zdrojový kód *AECalc (Azimuth & Elevation Calculation).*

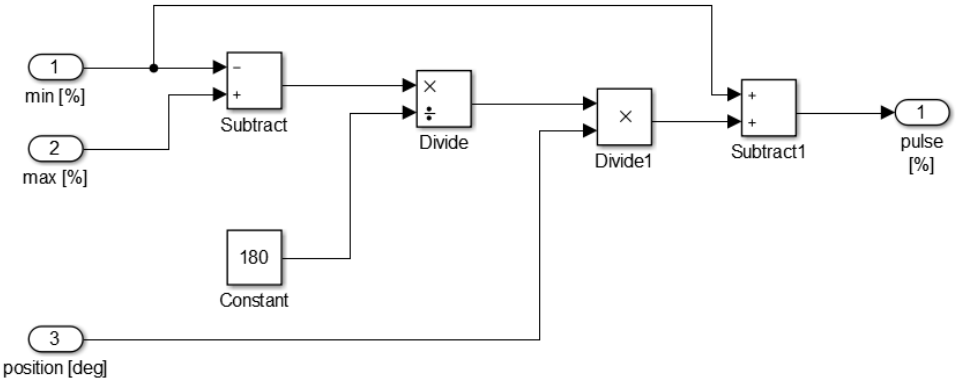
```
function [A_Out, E_Out] = AECalc(IsDet, CoordsArea, ResX, ResY,
Critdist, Fix, RView)
%{
Calculation of Azimuth & Elevation angles
-----------------------------------------------
IsDet ..... is detected an object?
CoordsArea ..... area of detected object
ResX, ResY ..... screen resolution
Critdist ..... critical distance for servo action
Fix ..... value in degrees for move servo
RView ..... reset view - go to starting position
A .... angle of azimuth - horizontal
E .... angle of elevation - vertical
scrCX, scrCY ..... center point of screen
-----------------------------------------------
%}
persistent A;
if isempty(A)
    A = 90;
end
persistent E;
if isempty(E)
    E = 90;
end

kvdr = 0;
gx = double(0);
gy = double(0);

% is detected and no reset view
if IsDet && ~RView
%center point of screen
scrCX = ResX / 2;
scrCY = ResY / 2;

% CoordsArea - vector [1x4]
x1 = CoordsArea(2);
y1 = CoordsArea(1);
w = CoordsArea(4);
h = CoordsArea(3);

% center of detected area (CoordsArea)
ObjCX = x1 + w / 2;
ObjCY = y1 + h / 2;

% -----------------------------------------------
% quadrant  I (top left)
if ObjCX > scrCX && ObjCY < scrCY
    kvdr = 1;
    gx = ResX - (x1 + w);
    gy = y1;
% quadrant II (top right)
elseif ObjCX < scrCX && ObjCY < scrCY
    kvdr = 2;
```

```
    gx = x1;
    gy = y1;
% quadrant III (down left)
elseif ObjCX < scrCX && ObjCY > scrCY
    kvdr = 3;
    gx = x1;
    gy = ResY - (y1 + h);
% quadrant IV (down right)
elseif ObjCX > scrCX && ObjCY > scrCY
    kvdr = 4;
    gx = ResX - (x1 + w);
    gy = ResY - (y1 + h);
end

% ---------------------------------------------
% comparing gap with Critdist - for servo action
if gx < Critdist
    % left
    if (kvdr == 2 || kvdr == 3) && (A + Fix <= 180)
        A = A + Fix;
    end
    % right
    if (kvdr == 1 || kvdr == 4) && (A - Fix >= 0)
        A = A - Fix;
    end
end
if gy < Critdist
    % up
    if (kvdr == 1 || kvdr == 2) && (E + Fix <= 180)
        E = E + Fix;
    end
    % down
    if (kvdr == 3 || kvdr == 4) && (E - Fix >= 45)
        E = E - Fix;
    end
end
% ~isDet ... return servos to start position
else
    A = 90;
    E = 90;
end

% reset view
if RView
    A = 90;
    E = 90;
end

A_Out = A;
E_Out = E;
```

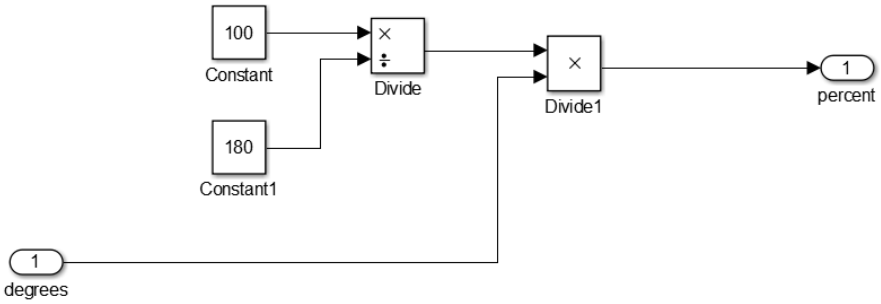
Příloha 14:

Př. 3) 3D polohování kamery

Subsystém *Interpolation to degrees*.



Subsystém *Convert degrees to percent*.



Příloha 15:
Př. 3) 3D polohování kamery

Zdrojový kód *PStop (Stop after positioning).*

```
function sampleOut = PStop(sampleIn, newPulse, SampleTime,
ServoTime100per)
%{
function for counting the min. time for servo move.
It depends on the angle of rotation, fcn counts samples
and terminate signal to servo after overtime.
-------------------------------------------------
newPulse ..... new pulse width value [%]
oldPulse ..... previous pulse width value [%]
SampleTime ..... time of 1 sample [s]
ServoTime100per ..... time required to rotate the servo min->max [s]
ServoSamp1per ..... count of samples for 1% of time move
-------------------------------------------------
%}
persistent oldPulse;
if isempty(oldPulse)
    oldPulse = 0;
end
persistent poc;
if isempty(poc)
    poc = 0;
end
persistent sampleDiff;
if isempty(sampleDiff)
    sampleDiff = 0;
end

% count all samples from generator
poc = poc + 1;

if newPulse ~= oldPulse
    ServoSamp1per = (1 / SampleTime * ServoTime100per) / 100;
    sampleDiff = abs(newPulse - oldPulse) * ServoSamp1per;
    oldPulse = newPulse;
    poc = 0;
else
    % counter is over and stops on max value
    if poc >= sampleDiff
        poc = sampleDiff;
    end
end

if poc < sampleDiff
    sampleOut = sampleIn;
else
    sampleOut = false;
end
```